

```
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGGGG  RRRRRRRRRRRR  TTTTTTTTTTTTTT  LLL
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGGGG  RRRRRRRRRRRR  TTTTTTTTTTTTTT  LLL
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGGGG  RRRRRRRRRRRR  TTTTTTTTTTTTTT  LLL
SSS            MMMMMM  MMMMMM  GGG            RRR      RRR      TTT      LLL
SSS            MMMMMM  MMMMMM  GGG            RRR      RRR      TTT      LLL
SSS            MMMMMM  MMMMMM  GGG            RRR      RRR      TTT      LLL
SSS            MMM      MMM      GGG            RRR      RRR      TTT      LLL
SSS            MMM      MMM      GGG            RRR      RRR      TTT      LLL
SSS            MMM      MMM      GGG            RRR      RRR      TTT      LLL
SSS            MMM      MMM      GGG            RRR      RRR      TTT      LLL
SSSSSSSSSSS    MMM      MMM      GGG            RRRRRRRRRRRR  TTT      LLL
SSSSSSSSSSS    MMM      MMM      GGG            RRRRRRRRRRRR  TTT      LLL
SSSSSSSSSSS    MMM      MMM      GGG            RRRRRRRRRRRR  TTT      LLL
SSS            MMM      MMM      GGG      GGGGGGGGGG  RRR      RRR      TTT      LLL
SSS            MMM      MMM      GGG      GGGGGGGGGG  RRR      RRR      TTT      LLL
SSS            MMM      MMM      GGG      GGGGGGGGGG  RRR      RRR      TTT      LLL
SSS            MMM      MMM      GGG      GGG      GGG  RRR      RRR      TTT      LLL
SSS            MMM      MMM      GGG      GGG      GGG  RRR      RRR      TTT      LLL
SSS            MMM      MMM      GGG      GGG      GGG  RRR      RRR      TTT      LLL
SSS            MMM      MMM      GGG      GGG      GGG  RRR      RRR      TTT      LLL
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGG      RRR      RRR      TTT      LLLLLLLLLLLLLLLL
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGG      RRR      RRR      TTT      LLLLLLLLLLLLLLLL
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGG      RRR      RRR      TTT      LLLLLLLLLLLLLLLL
```

```
SSSSSSSS MM MM GGGGGGGG DDDDDDDD IIIIII SSSSSSSS LL IIIIII NN NN
SSSSSSSS MM MM GGGGGGGG DDDDDDDD IIIIII SSSSSSSS LL IIIIII NN NN
SS SS M M M M GG GG DD DD II II SS SS LL LL NN NN
SS SS M M M M GG GG DD DD II II SS SS LL LL NN NN
SS SSSSSS MM MM GG GG DD DD II II SS SSSSSS LL LL NN NN
SS SSSSSS MM MM GG GG DD DD II II SS SSSSSS LL LL NN NN
SS SS MM MM GG GGGGGG DD DD II II SS SS LL LL NN NN
SS SS MM MM GG GGGGGG DD DD II II SS SS LL LL NN NN
SS SS MM MM GG GG DD DD II II SS SS LL LL NN NN
SSSSSSSS MM MM GGGGGG DDDDDDDD IIIIII SSSSSSSS LLLLLLLLLL IIIIII NN NN
SSSSSSSS MM MM GGGGGG DDDDDDDD IIIIII SSSSSSSS LLLLLLLLLL IIIIII NN NN
...
LL IIIIII SSSSSSSS
LL IIIIII SS
LL II SS
LL II SS
LL II SS
LL II SSSSSS
LL II SSSSSS SS
LL II SS
LL II SS
LL II SS
LLLLLLLLLLLL IIIIII SSSSSSSS
LLLLLLLLLLLL IIIIII SSSSSSSS
```



```
0001 0 %TITLE 'SMG$DISPLAY LINKS - Virtual Display Linkages'
0002 0 MODULE SMG$DISPLAY LINKS (
0003 0 IDENT = '1-096' ! File: SMGDISLIN.B32 Edit: STAN1096
0004 0 ) =
0005 1 BEGIN
0006 1
0007 1 *****
0008 1 *
0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0011 1 * ALL RIGHTS RESERVED.
0012 1 *
0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0018 1 * TRANSFERRED.
0019 1 *
0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0022 1 * CORPORATION.
0023 1 *
0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0026 1 *
0027 1 *
0028 1 *****
0029 1
0030 1
0031 1 ++
0032 1 FACILITY:      Screen Management
0033 1
0034 1 ABSTRACT:
0035 1 The procedures in this module are concerned only with the
0036 1 allocation/deallocation of virtual displays, and with the pasting/
0037 1 unpasting of these virtual displays to pasteboards. The are not
0038 1 concerned with their contents or output.
0039 1
0040 1 For the procedures which maintain and update the contents of
0041 1 virtual displays, see the module SMG$DISPLAY_CHANGE.
0042 1
0043 1 For the procedures which actually do output from these virtual
0044 1 displays, see the module SMG$DISPLAY_OUTPUT.
0045 1
0046 1 For procedures that support input operations, see the module
0047 1 SMG$DISPLAY_INPUT.
0048 1
0049 1 ENVIRONMENT: User mode, Shared library routines.
0050 1
0051 1 AUTHOR: R. Reichert, CREATION DATE: 26-Jan-1983
0052 1
0053 1 MODIFIED BY:
0054 1
0055 1 1-096 - Don't allow paste or unpaste if display is batched.
0056 1 STAN 27-Jun-1984.
0057 1 1-095 - Use symbolic names SMG$K_TOP, etc. in SMG$LABEL_BORDER.
```

SMG\$DISPLAY\_LIN SMG\$DISPLAY\_LINKS - Virtual Display Linkages  
1-096

L 11  
16-Sep-1984 00:29:22  
14-Sep-1984 13:09:43

VAX-11 Bliss-32 V4.0-742  
[SMGRTL.SRC]SMGDISLIN.B32;1

Page 2  
(1)

```
: 58      0058 1 | Change error messages by SMG$SET_DISPLAY_SCROLLING_REGION.  
: 59      0059 1 | STAN 3-Jun-1984.  
: 60      0060 1 | 1-094 - Fix bug re borders occluding other borders. STAN 7-May-1984.  
: 61      0061 1 | 1-001 - Original. Skeleton for future code. RKR 26-Jan-1983  
: 62      0062 1 | --
```



```
64      0063 1 %SBTTL 'Declarations'
65      0064 1
66      0065 1 SWITCHES:
67      0066 1
68      0067 1
69      0068 1
70      0069 1 LINKAGES:
71      0070 1
72      0071 1 NONE
73      0072 1
74      0073 1 INCLUDE FILES
75      0074 1
76      0075 1
77      0076 1 REQUIRE 'RTLIN:SMGPROLOG';      ! defines psects, macros, tcb,
78      0154 1      ! wcb, & terminal symbols
79      0155 1
80      0156 1 REQUIRE 'RTLIN:STRLNK';      ! Linkages to string JSB
81      0341 1
82      0342 1
83      0343 1 TABLE OF CONTENTS:
84      0344 1
85      0345 1
86      0346 1 FORWARD ROUTINE
87      0347 1
88      0348 1 ! Public entry points
89      0349 1
90      0350 1 SMG$CHANGE_PBD_CHARACTERISTICS,      ! Change characteristics of
91      0351 1      ! physical terminal
92      0352 1
93      0353 1 SMG$CHANGE_VIRTUAL_DISPLAY,      ! Change characteristics of
94      0354 1      ! existing virtual display
95      0355 1
96      0356 1 SMG$CHECK_FOR_OCCLUSION,      ! Check to see if a virtual
97      0357 1      ! display is occluded.
98      0358 1
99      0359 1 SMG$CREATE_PASTEBOARD,      ! Create pasteboard
100     0360 1
101     0361 1 SMG$CREATE_VIRTUAL_DISPLAY,      ! Create virtual display
102     0362 1
103     0363 1 SMG$DELETE_PASTEBOARD,      ! Get rid of pasteboard, terminate
104     0364 1      ! all operations on this display
105     0365 1
106     0366 1 SMG$DELETE_VIRTUAL_DISPLAY,      ! Delete virtual display
107     0367 1
108     0368 1 SMG$GET_DISPLAY_ATTR,      ! Return current attributes of
109     0369 1      ! virtual display
110     0370 1
111     0371 1 SMG$LABEL_BORDER,      ! Supply label for border
112     0372 1
113     0373 1 SMG$MOVE_VIRTUAL_DISPLAY,      ! Move position of virtual
114     0374 1      ! display on pasteboard
115     0375 1
116     0376 1 SMG$PASTE_VIRTUAL_DISPLAY,      ! Paste virtual display to
117     0377 1      ! pasteboard
118     0378 1
119     0379 1 SMG$POP_VIRTUAL_DISPLAY,      ! Pop off (and delete) all
120     0380 1      ! virtual displays from given
```

```

: 121      0381 1
: 122      0382 1
: 123      0383 1      SMG$REPASTE_VIRTUAL_DISPLAY,
: 124      0384 1
: 125      0385 1
: 126      0386 1      SMG$RESTORE_PHYSICAL_SCREEN,
: 127      0387 1
: 128      0388 1
: 129      0389 1
: 130      0390 1      SMG$SAVE_PHYSICAL_SCREEN,
: 131      0391 1
: 132      0392 1
: 133      0393 1      SMG$SET_DISPLAY_SCROLL_REGION,
: 134      0394 1
: 135      0395 1
: 136      0396 1      SMG$UNPASTE_VIRTUAL_DISPLAY,
: 137      0397 1
: 138      0398 1
: 139      0399 1      ! Private entry points
: 140      0400 1
: 141      0401 1      SMG$$CALC_PASTE_TRANSF,
: 142      0402 1
: 143      0403 1
: 144      0404 1      SMG$$CHECK_OCCLUSION,
: 145      0405 1
: 146      0406 1
: 147      0407 1
: 148      0408 1      SMG$$CHECK_OCCLUSION_FIRST,
: 149      0409 1
: 150      0410 1
: 151      0411 1      SMG$$CREATE_PASTEBOARD,
: 152      0412 1
: 153      0413 1      SMG$$CREATE_VIRTUAL_DISPLAY,
: 154      0414 1
: 155      0415 1
: 156      0416 1      SMG$$CREATE_WCB,
: 157      0417 1
: 158      0418 1      SMG$$DEALLOCATE_WCB,
: 159      0419 1
: 160      0420 1      SMG$$DUPL_VIRTUAL_DISPLAY,
: 161      0421 1
: 162      0422 1      SMG$$LOCATE_PP,
: 163      0423 1
: 164      0424 1
: 165      0425 1      SMG$$PASTE_VIRTUAL_DISPLAY,
: 166      0426 1
: 167      0427 1
: 168      0428 1      SMG$$RECALC_PP_FIELDS,
: 169      0429 1
: 170      0430 1
: 171      0431 1      SMG$$UNPASTE_VIRTUAL_DISPLAY;
: 172      0432 1
: 173      0433 1
: 174      0434 1
: 175      0435 1
: 176      0436 1
: 177      0437 1      !

```

```

! to top of pasting stack.
! Repaste virtual display to
! pasteboard in new position
! Restore screen to where it
! was after non-SMG user
! munged it.
! Save physical screen before
! non-SMG user mungs its up.
! Set the scrolling region in
! a virtual display
! Unpaste virtual display from
! pasteboard.
! Calculate pasting
! transformation constants.
! Check current complement of
! pasted virtual displays to
! see who is occluded.
! Check occlusion caused by
! highest pasted virtual display.
! Create pasteboard
! Inner-most Create Virtual
! Display routine
! Create WCB and its buffers
! Get rid of WCB and its buffers.
! Duplicate a virtual display
! Locate PP which matches a
! DCB and a PBCB.
! Inner-most Paste Virtual
! Display routine.
! Recalculate pasting packet
! fields after virtual display
! batching ceases.
! Inner-most Unpaste Virtual
! Display routine.
! routines.

```



```

: 178      0438 1 ! EXTERNAL REFERENCES
: 179      0439 1 !
: 180      0440 1 EXTERNAL ROUTINE
: 181      0441 1   LIB$ANALYZE_SDESC_R2 : LIB$ANALYZE_SDESC_JSB_LINK,
: 182      0442 1   ! Get length and address of a string
: 183      0443 1
: 184      0444 1   LIB$FREE_VM, ! Deallocate heap storage
: 185      0445 1
: 186      0446 1   LIB$FREE_EF, ! Free an event flag
: 187      0447 1
: 188      0448 1   LIB$GET_EF, ! Get an event flag
: 189      0449 1
: 190      0450 1   LIB$GET_VM, ! Allocate heap storage
: 191      0451 1
: 192      0452 1   LIB$COPY_DXDX, ! String copy by descriptor
: 193      0453 1
: 194      0454 1   LIB$FREE1_DD, ! Free a dynamic string
: 195      0455 1
: 196      0456 1   SMG$$BEGIN_PASTEBOARD_UPDATE_R1 : SMG$$BEGIN_PBD_UPDATE$LNK,
: 197      0457 1   ! Increase buffering level by 1
: 198      0458 1
: 199      0459 1   SMG$$END_PASTEBOARD_UPDATE_R2 : SMG$$END_PBD_UPDATE$LNK,
: 200      0460 1   ! Decrease buffering level by 1
: 201      0461 1
: 202      0462 1   SMG$$ERASE_PASTEBOARD, ! Erase the physical screen
: 203      0463 1
: 204      0464 1   SMG$$CHECK_FOR_OUTPUT_DCB, ! Force output if now is the time
: 205      0465 1
: 206      0466 1   SMG$$CHECK_FOR_OUTPUT_PBCB, ! Force output
: 207      0467 1
: 208      0468 1   SMG$$FILL_WINDOW_BUFFER, ! Move stuff from virt. display to
: 209      0469 1   ! pasteboard buffer and output.
: 210      0470 1
: 211      0471 1   SMG$$FIND_MIN_CURSOR_POS, ! Set cursor on physical screen
: 212      0472 1   SMG$$FLUSH_BUFFER, ! Flush output buffer
: 213      0473 1
: 214      0474 1   SMG$$FORCE_SCROLL_REG, ! Force scrolling region on screen.
: 215      0475 1
: 216      0476 1   SMG$$OUTPUT, ! Output a string to terminal
: 217      0477 1
: 218      0478 1   SMG$$OCCLUDE, ! Check for how two rectangular areas
: 219      0479 1   ! overlap.
: 220      0480 1
: 221      0481 1   SMG$$PBCB_EXIT_HANDLER, ! Output exit handler
: 222      0482 1
: 223      0483 1   SMG$$SETUP_TERMINAL_TYPE; ! Get device characteristics
: 224      0484 1
: 225      0485 1 EXTERNAL LITERAL
: 226      0486 1
: 227      0487 1   LIB$EF_ALRFRE, ! Event flag already free
: 228      0488 1   SMG$_BATWAS_ON, ! Batching was enabled
: 229      0489 1   SMG$_FATERRCIB, ! Fatal error in library
: 230      0490 1   SMG$_INVARG, ! Invalid argument
: 231      0491 1   SMG$_ILLBATFNC, ! Operation not legal to batched display
: 232      0492 1   SMG$_INVDIS_ID, ! Invalid virtual display id
: 233      0493 1   SMG$_INVPAS_ID, ! Invalid pasteboard id
: 234      0494 1   SMG$_INVROW, ! Invalid row

```



SMG\$DISPLAY\_LIN SMG\$DISPLAY\_LINKS - Virtual Display Linkages  
1-096 Declarations

C 12  
16-Sep-1984 00:29:22  
14-Sep-1984 13:09:43

VAX-11 Bliss-32 V4.0-742  
[SMGRTL.SRC]SMGDISLIN.B32;1

Page 6  
(2)

:	235	0495	1	SMG\$_NOTPASTED,	:	Given display is not pasted to given
:	236	0496	1		:	pasteboard
:	237	0497	1	SMG\$_PASALREXI,	:	Pasteboard already exists for this device
:	238	0498	1	SMG\$_TOOMANDIS,	:	Too many virtual displays requested
:	239	0499	1	SMG\$_TOOMANPAS,	:	Too many pasteboards requested
:	240	0500	1	SMG\$_WRONUMARG;	:	Wrong number of arguments

SMG  
1-0



```

: 242      0501 1  !+
: 243      0502 1  Pasteboard Directory (PBD)
: 244      0503 1  -----
: 245      0504 1  This data structure resides in OWN storage. It is the primary vehicle
: 246      0505 1  for getting from a pasteboard id to the associated pasteboard control
: 247      0506 1  block.
: 248      0507 1  -
: 249      0508 1
: 250      0509 1 GLOBAL
: 251      0510 1   PBD_L_COUNT : INITIAL (0), ! No. of pasteboards we currently know
: 252      0511 1   ! about.
: 253      0512 1
: 254      0513 1   PBD_A_PBCB : VECTOR [PBD_K_MAX_PB, LONG]
: 255      0514 1   INITIAL (REP PBD_K_MAX_PB OF (0)),
: 256      0515 1   ! List of pasteboard addresses. Indexed by
: 257      0516 1   ! pasteboard id (PID) to find address of
: 258      0517 1   ! corresponding PBCB.
: 259      0518 1
: 260      0519 1   PBD_V_PB_AVAIL : BITVECTOR [PBD_K_MAX_PB]
: 261      0520 1   INITIAL ( BYTE (REP ((PBD_K_MAX_PB+7)/8) OF (0)));
: 262      0521 1   ! This is a bit-vector of pasteboard id's
: 263      0522 1   ! still available. The next available number
: 264      0523 1   ! is found by doing a FFC instruction to find
: 265      0524 1   ! first bit which is a 0. The bit position so
: 266      0525 1   ! computed is the next available PID. The
: 267      0526 1   ! bit found is set to 1 to mark it as in use.
: 268      0527 1   ! (Presumably, a check has already been made to
: 269      0528 1   ! insure that PBD_L_COUNT is LSS PBD_K_MAX_PB.)
: 270      0529 1
: 271      0530 1 ! Some constants needed by reference for FFC instruction
: 272      0531 1 OWN
: 273      0532 1   ZERO
: 274      0533 1   PBD_K_MAX_PB_BY_REF : INITIAL ( PBD_K_MAX_PB );

```



```

: 276      0534 1 %SBTTL 'SMG$CHANGE_PBD_CHARACTERISTICS'
: 277      0535 1 GLOBAL ROUTINE SMG$CHANGE_PBD_CHARACTERISTICS
: 278      0536 1      (PBID,
: 279      0537 1          P_DESIRED_WIDTH,
: 280      0538 1          P_RESULTING_WIDTH,
: 281      0539 1          P_DESIRED_HEIGHT,
: 282      0540 1          P_RESULTING_HEIGHT,
: 283      0541 1          P_DESIRED_BACKGROUND_COLOR,
: 284      0542 1          P_RESULTING_BACKGROUND_COLOR
: 285      0543 1      )=
: 286      0544 1
: 287      0545 1 ++
: 288      0546 1 FUNCTIONAL DESCRIPTION:
: 289      0547 1
: 290      0548 1      This routine lets you change the physical dimensions
: 291      0549 1      of a pasteboard. It also lets you change the background color.
: 292      0550 1
: 293      0551 1 CALLING SEQUENCE:
: 294      0552 1
: 295      0553 1      ret_status.wlc.v = SMG$CHANGE_PBD_CHARACTERISTICS
: 296      0554 1      (PBID.rl.r
: 297      0555 1          [,DESIRED_WIDTH.rl.r]
: 298      0556 1          [,RESULTING_WIDTH.wl.r]
: 299      0557 1          [,DESIRED_HEIGHT.rl.r]
: 300      0558 1          [,RESULTING_HEIGHT.wl.r]
: 301      0559 1          [,DESIRED_BACKGROUND_COLOR.rl.r]
: 302      0560 1          [,RESULTING_BACKGROUND_COLOR.wl.r]
: 303      0561 1      )
: 304      0562 1
: 305      0563 1 FORMAL PARAMETERS:
: 306      0564 1
: 307      0565 1      PBID.rl.r      Pasteboard id of pasteboard.
: 308      0566 1
: 309      0567 1      DESIRED_WIDTH.rl.r      New width desired for pasteboard.
: 310      0568 1      If omitted, the width is not changed.
: 311      0569 1
: 312      0570 1      RESULTING_WIDTH.wl.r      Physical width that resulted. This may
: 313      0571 1      be larger than the width requested if the
: 314      0572 1      terminal width couldn't be set exactly to
: 315      0573 1      the desired width. This may be smaller
: 316      0574 1      than the width requested if the terminal
: 317      0575 1      width couldn't be set that wide.
: 318      0576 1      In this case, the terminal was set to
: 319      0577 1      it's maximum width.
: 320      0578 1
: 321      0579 1      Example:      (for VT100)
: 322      0580 1
: 323      0581 1      Width Desired      Width resulting
: 324      0582 1
: 325      0583 1      60      80
: 326      0584 1      110      132
: 327      0585 1      150      132
: 328      0586 1
: 329      0587 1      If desired width was omitted, this
: 330      0588 1      argument receives the current pasteboard
: 331      0589 1      width.
: 332      0590 1      To find out what the pasteboard width is

```



```

333      0591 1
334      0592 1
335      0593 1
336      0594 1
337      0595 1
338      0596 1
339      0597 1
340      0598 1
341      0599 1
342      0600 1
343      0601 1
344      0602 1
345      0603 1
346      0604 1
347      0605 1
348      0606 1
349      0607 1
350      0608 1
351      0609 1
352      0610 1
353      0611 1
354      0612 1
355      0613 1
356      0614 1
357      0615 1
358      0616 1
359      0617 1
360      0618 1
361      0619 1
362      0620 1
363      0621 1
364      0622 1
365      0623 1
366      0624 1
367      0625 1
368      0626 1
369      0627 1
370      0628 1
371      0629 1
372      0630 1
373      0631 1
374      0632 1
375      0633 1
376      0634 1
377      0635 1
378      0636 1
379      0637 1
380      0638 1
381      0639 1
382      0640 1
383      0641 1
384      0642 1
385      0643 1
386      0644 1
387      0645 1
388      0646 1
389      0647 1

```

(as opposed to the terminal width),  
the caller should take the minimum  
of his desired width and the resulting width.

DESIRED\_HEIGHT.rl.r New height desired for pasteboard.  
If omitted, the height is not changed.

RESULTING\_HEIGHT.wl.r Physical height that resulted. This may  
be larger than the height requested if the  
terminal height couldn't be set exactly to  
the desired height. This may be smaller  
than the height requested if the terminal  
height couldn't be set that high.  
In this case, the terminal was set to  
it's maximum height.

Example: (for VT100)

Height Desired	Height resulting
15	24
35	24

To find out what the pasteboard height is  
(as opposed to the terminal height),  
the caller should take the minimum  
of his desired height and the resulting height.

DESIRED\_BACKGROUND\_COLOR.rl.r Symbolic name for the background  
color wanted. For example,  
SMG\$C\_COLOR\_WHITE. These symbols  
are defined in SMGDEF.SDL.  
If omitted, the background color  
is not changed.

RESULTING\_BACKGROUND\_COLOR.wl.r Receives the actual background color  
that was chosen. If the terminal  
does not support the exact color  
desired, the nearest approximation  
will be chosen. This is determined  
by comparing the frequency of the  
desired light wave against the  
available frequencies. For more  
information about colorimetry,  
consult National Bureau of Standards  
Circular 553, The ISCC-NBS method of  
designating Colors.

Example: (VT100)

Color desired	Resulting Color
yellowish pink	white
navy blue	black



```

: 390      0648 1 |
: 391      0649 1 |
: 392      0650 1 |
: 393      0651 1 |
: 394      0652 1 | IMPLICIT INPUTS:
: 395      0653 1 |
: 396      0654 1 |     NONE
: 397      0655 1 |
: 398      0656 1 | IMPLICIT OUTPUTS:
: 399      0657 1 |
: 400      0658 1 |     NONE
: 401      0659 1 |
: 402      0660 1 | COMPLETION STATUS:
: 403      0661 1 |
: 404      0662 1 |     SS$ NORMAL      Normal successful completion
: 405      0663 1 |     SMG$_WRONUMARG  Wrong number of arguments.
: 406      0664 1 |     SMG$_PBDIN_USE  Can't change characteristics while buffering is on
: 407      0665 1 |     SMG$_INVWIDARG  Invalid width of 0 desired
: 408      0666 1 |     SMG$_INVPAGARG  Invalid height of 0 desired
: 409      0667 1 |     SMG$_INVCOLARG  Unknown background color specified
: 410      0668 1 |
: 411      0669 1 | SIDE EFFECTS:
: 412      0670 1 |
: 413      0671 1 |     Physical width and background color of terminal may change.
: 414      0672 1 | --

```

If the desired color is omitted,  
the value of this variable is not  
affected.



```

: 416      0673  2 BEGIN
: 417      0674  2
: 418      0675  2 EXTERNAL ROUTINE
: 419      0676  2
: 420      0677  2     SMG$$CHECK FOR OUTPUT_PBCB,
: 421      0678  2     SMG$$CALC_PASTE_TRANSF,
: 422      0679  2     SMG$$CREATE_WCB,
: 423      0680  2     SMG$$DEALLOCATE_WCB,
: 424      0681  2     SMG$$ERASE_PASTEBOARD,
: 425      0682  2     SMG$$OUTPUT;
: 426      0683  2
: 427      0684  2 EXTERNAL LITERAL
: 428      0685  2
: 429      0686  2     SMG$_INVWIDARG,      ! width=0
: 430      0687  2     SMG$_INVPAGARG,      ! HEIGHT=0
: 431      0688  2     SMG$_INVCOLARG,      ! unknown color
: 432      0689  2     SMG$_PBDIN_USE;      ! pasteboard was batched
: 433      0690  2
: 434      0691  2 BUILTIN
: 435      0692  2
: 436      0693  2     NULLPARAMETER;
: 437      0694  2
: 438      0695  2 LOCAL
: 439      0696  2
: 440      0697  2     STATUS,                ! Status of subroutine calls
: 441      0698  2     PASTING_PACKET_PANIC,    ! TRUE if we must adjust pasting packets
: 442      0699  2     CURR_PP: REF $PP_DECL,    ! Pasting packet pointer
: 443      0700  2
: 444      0701  2     PBCB      : REF $PBCB_DECL; ! Address of pasteboard control block

```

```

446 0702 2 $SMG$VALIDATE_ARGCOUNT (1, 7); ! Test for right no. of args
447 0703
448 0704 $SMG$GET_PBCB (.PBCB,PBCB); ! Get address of PBCB
449 0705
450 0706 PASTING_PACKET_PANIC=0;
451 0707
452 0708 !+
453 0709 ! If a desired width is specified, get it now.
454 0710 !-
455 0711
456 0712 IF NOT NULLPARAMETER(P_DESIRED_WIDTH)
457 0713 THEN BEGIN ! Change pasteboard width
458 0714 BIND DESIRED_WIDTH=.P_DESIRED_WIDTH;
459 0715 ! (a) LOCAL CURRENT_MAX, DESIRED_MAX;
460 0716 LOCAL PREVIOUS_WIDTH;
461 0717 LOCAL RESULTANT_WIDTH;
462 0718
463 0719 IF .DESIRED_WIDTH EQL 0
464 0720 THEN RETURN SMG$_INVWIDARG;
465 0721
466 0722 !+
467 0723 ! Determine the physical setting of the terminal by rounding
468 0724 ! up to 80 or 132 as necessary. Do the same for the desired
469 0725 ! width. Compare these two numbers to see if we must change
470 0726 ! the width. This algorithm will have to change if we ever
471 0727 ! support terminals with widths other than 80 and 132.
472 0728 !-
473 0729
474 0730 IF .PBCB[PBCB L_BATCH_LEVEL] NEQ 0
475 0731 THEN RETURN SMG$_PBDIN USE;
476 0732 ! (a) IF .PBCB[PBCB W_WIDTH] LEQ 80
477 0733 ! (a) THEN CURRENT_MAX=80
478 0734 ! (a) ELSE CURRENT_MAX=132;
479 0735 ! (a) IF .DESIRED_WIDTH LEQ 80
480 0736 ! (a) THEN DESIRED_MAX=80
481 0737 ! (a) ELSE DESIRED_MAX=132;
482 0738
483 0739 ! (a) !+
484 0740 ! (a) ! If the desired max is the same as the current max,
485 0741 ! (a) ! then no escape sequence need be sent to the terminal.
486 0742 ! (a) ! Just adjust our internal width in the PBCB.
487 0743 ! (a) !-
488 0744
489 0745 ! (a) IF .DESIRED_MAX NEQ .CURRENT_MAX
490 0746 ! (a) THEN
491 0747
492 0748 !+
493 0749
494 0750 Note: (a)
495 0751
496 0752 The lines marked !(a) could be added back in
497 0753 if you want to avoid outputting the escape sequence
498 0754 to change the terminal width if it isn't necessary.
499 0755 However, that will mean the screen doesn't physically
500 0756 blank and so extra code would have to be written to
501 0757 blank the right part of a screen when changing width
502 0758 (say) from 70 to 50 columns.
```



```

503      0759 3 !-
504      0760 3
505      0761 4 BEGIN ! Change physical width
506      0762 4
507      0763 4 LOCAL
508      0764 4
509      0765 4 NORMAL_WIDTH,
510      0766 4 WIDE_WIDTH;
511      0767 4
512      0768 4 !+
513      0769 4 ! First, clear the screen.
514      0770 4 !-
515      0771 4
516      0772 4 $SMG$GET_TERM_DATA(ERASE_WHOLE_DISPLAY);
517      0773 4 IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
518      0774 5 THEN BEGIN
519      0775 5 STATUS = SMG$$OUTPUT(.PBCB,.PBCB[PBCB_L_CAP_LENGTH],
520      0776 5 .PBCB[PBCB_A_CAP_BUFFER]);
521      0777 5 IF NOT .STATUS THEN RETURN .STATUS
522      0778 4 END;
523      0779 4
524      0780 4 !+
525      0781 4 ! Second, get the normal size.
526      0782 4 !-
527      0783 4
528      0784 4 $SMG$GET_TERM_DATA(COLUMNS);
529      0785 4 IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
530      0786 5 THEN BEGIN
531      0787 5 BIND RESULT=.PBCB[PBCB_A_CAP_BUFFER];
532      0788 5 STATUS = SMG$$OUTPUT(.PBCB,.PBCB[PBCB_L_CAP_LENGTH],
533      0789 5 .PBCB[PBCB_A_CAP_BUFFER]);
534      0790 5 IF NOT .STATUS THEN RETURN .STATUS;
535      0791 5 NORMAL_WIDTH=.RESULT
536      0792 5 END
537      0793 4 ELSE NORMAL_WIDTH=80;
538      0794 4
539      0795 4 !+
540      0796 4 ! Third, get the wide size.
541      0797 4 !-
542      0798 4
543      0799 4 $SMG$GET_TERM_DATA(WIDTH WIDE);
544      0800 4 IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
545      0801 5 THEN BEGIN
546      0802 5 BIND RESULT=.PBCB[PBCB_A_CAP_BUFFER];
547      0803 5 STATUS = SMG$$OUTPUT(.PBCB,.PBCB[PBCB_L_CAP_LENGTH],
548      0804 5 .PBCB[PBCB_A_CAP_BUFFER]);
549      0805 5 IF NOT .STATUS THEN RETURN .STATUS;
550      0806 5 WIDE_WIDTH=.RESULT
551      0807 5 END
552      0808 4 ELSE WIDE_WIDTH=80;
553      0809 4
554      0810 4 !+
555      0811 4 ! Decide which sequence to send.
556      0812 4 !-
557      0813 4
558      0814 4 IF .DESIRED_WIDTH LEQ .NORMAL_WIDTH
559      0815 5 THEN BEGIN

```

```
560      0816 5      $SMG$GET_TERM_DATA(WIDTH NARROW);
561      0817 5      RESULTANT_WIDTH=.NORMAL_WIDTH
562      0818 5      END
563      0819 5      ELSE BEGIN
564      0820 5      $SMG$GET_TERM_DATA(WIDTH WIDE);
565      0821 5      RESULTANT_WIDTH=.WIDE_WIDTH
566      0822 4      END;
567      0823 4
568      0824 4      IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
569      0825 5      THEN BEGIN
570      0826 5          STATUS = SMG$$OUTPUT(.PBCB,.PBCB[PBCB_L_CAP_LENGTH],
571      0827 5              .PBCB[PBCB_A_CAP_BUFFER]);
572      0828 5          IF NOT .STATUS THEN RETURN .STATUS;
573      0829 4          END;
574      0830 4
575      0831 4      !+
576      0832 4      If we asked for something smaller than the terminal
577      0833 4      could handle (like a width of 60 on an 80-column terminal)
578      0834 4      then we will software simulate the smaller width.
579      0835 4      -
580      0836 4
581      0837 4      RESULTANT_WIDTH=MINU(.RESULTANT_WIDTH,.DESIRED_WIDTH);
582      0838 4
583      0839 3      END;      ! Change physical width
584      0840 3
585      0841 3      !+
586      0842 3      Should we go back to the old scheme whereby we
587      0843 3      output the escape sequence only if the max width has
588      0844 3      changed, then we need the following line:
589      0845 3
590      0846 3      (a) ELSE      RESULTANT_WIDTH=.DESIRED_WIDTH;
591      0847 3      -
592      0848 3
593      0849 3      !+
594      0850 3      Save away new pasteboard width in the PBCB.
595      0851 3      -
596      0852 3
597      0853 3      PREVIOUS_WIDTH=.PBCB[PBCB_W_WIDTH];
598      0854 3      PBCB[PBCB_W_WIDTH]=.RESULTANT_WIDTH;
599      0855 3
600      0856 3      !+
601      0857 3      If the width changed, we must recalculate all the pasting
602      0858 3      packet parameters pronto. Make a note.
603      0859 3      -
604      0860 3
605      0861 3      IF .PREVIOUS_WIDTH NEQ .RESULTANT_WIDTH
606      0862 3      THEN PASTING_PACKET_PANIC=1;
607      0863 3      PASTING_PACKET_PANIC=1;
608      0864 3
609      0865 3      !+
610      0866 3      At some point in the future, we might want to tell VMS
611      0867 3      about this new width. If so, we would add that code here.
612      0868 3      There is probably no need to do that since we will restore
613      0869 3      the original width when we delete this pasteboard.
614      0870 3      -
615      0871 3
616      0872 2      END;      ! Change pasteboard width
```



```

: 617      0873      2
: 618      0874      2 !+
: 619      0875      2 !- If the user wants the pasteboard width, give it to him now.
: 620      0876      2 !-
: 621      0877      2
: 622      0878      2 IF NOT NULLPARAMETER(P_RESULTING_WIDTH)
: 623      0879      3 THEN BEGIN ! Return pasteboard width
: 624      0880      3 BIND RESULTING_WIDTH = .P_RESULTING_WIDTH;
: 625      0881      3 RESULTING_WIDTH=.PBCB[PBCB_W_WIDTH]
: 626      0882      2 END; ! Return pasteboard width
: 627      0883      2
: 628      0884      2 !+
: 629      0885      2 !- If the user wants to change his height, do that now.
: 630      0886      2 !- If he specifies an illegal height, that's his problem;
: 631      0887      2 !- we don't know what sort of funny terminal he might have.
: 632      0888      2 !- This code will have to change if we ever support terminals
: 633      0889      2 !- that can change height by sending them escape sequences.
: 634      0890      2 !-
: 635      0891      2
: 636      0892      2 IF NOT NULLPARAMETER(P_DESIRED_HEIGHT)
: 637      0893      3 THEN BEGIN ! Change pasteboard height
: 638      0894      3 BIND DESIRED_HEIGHT = .P_DESIRED_HEIGHT;
: 639      0895      3 IF .PBCB[PBCB_L_BATCH_LEVEL] NEQ 0
: 640      0896      3 THEN RETURN SMG$PBDIN_USE;
: 641      0897      3 IF .DESIRED_HEIGHT EQL 0
: 642      0898      3 THEN RETURN SMG$INVPAGARG;
: 643      0899      3 IF .PBCB[PBCB_B_ROWS] NEQ .DESIRED_HEIGHT
: 644      0900      4 THEN BEGIN
: 645      0901      4 !+
: 646      0902      4 !- Blank screen if we are making screen smaller,
: 647      0903      4 !- so as to get rid of items after the bottom of
: 648      0904      4 !- the pasteboard.
: 649      0905      4 !-
: 650      0906      4 IF MINU(24,.DESIRED_HEIGHT) LSSU .PBCB[PBCB_B_ROWS]
: 651      0907      5 THEN BEGIN
: 652      0908      5 STATUS=SMG$$ERASE PASTEBOARD(.PBCB);
: 653      0909      5 IF NOT .STATUS THEN RETURN .STATUS
: 654      0910      4 END;
: 655      0911      4 !+
: 656      0912      4 !- All existing terminals have a maximum height of 24.
: 657      0913      4 !-
: 658      0914      4 PBCB[PBCB_B_ROWS]=MINU(24,.DESIRED_HEIGHT);
: 659      0915      4 PASTING_PACRET_PANIC=1
: 660      0916      4 END
: 661      0917      2 END; ! Change pasteboard height
: 662      0918      2
: 663      0919      2 !+
: 664      0920      2 !- If the user wants the pasteboard height, give it to him now.
: 665      0921      2 !-
: 666      0922      2
: 667      0923      2 IF NOT NULLPARAMETER(P_RESULTING_HEIGHT)
: 668      0924      3 THEN BEGIN ! Return pasteboard height
: 669      0925      3 BIND RESULTING_HEIGHT=.P_RESULTING_HEIGHT;
: 670      0926      3 RESULTING_HEIGHT=.PBCB[PBCB_B_ROWS]
: 671      0927      2 END; ! Return pasteboard height
: 672      0928      2
: 673      0929      2 !+

```



```

: 674      0930 2  ! If we changed either the width or height of the pasteboard,
: 675      0931 2  ! then we must go adjust all the pasting packets now.
: 676      0932 2  ! We must also reallocate and reshape the buffers in the WCB.
: 677      0933 2  !-
: 678      0934 2  !-
: 679      0935 2  IF .PASTING_PACKET_PANIC
: 680      0936 2  THEN BEGIN ! Update all pasting packets
: 681      0937 2  LOCAL CURR_PP : REF $PP_DECL;
: 682      0938 2  +
: 683      0939 2  ! Deallocate the old WCB.
: 684      0940 2  !-
: 685      0941 2  STATUS=SMG$$DEALLOCATE_WCB(.PBCB[PBCB_A_WCB]);
: 686      0942 2  IF NOT .STATUS THEN RETURN .STATUS;
: 687      0943 2  +
: 688      0944 2  ! Allocate a new WCB.
: 689      0945 2  !-
: 690      0946 2  STATUS=SMG$$CREATE_WCB( %REF(.PBCB[PBCB_B_ROWS]),
: 691      0947 2  %REF(.PBCB[PBCB_W_WIDTH]),
: 692      0948 2  PBCB[PBCB_A_WCB]);
: 693      0949 2  IF NOT .STATUS THEN RETURN .STATUS;
: 694      0950 2  +
: 695      0951 2  ! Walk chain of DCB's for all displays currently pasted
: 696      0952 2  ! to this pasteboard, and go update their pasting packet.
: 697      0953 2  ! Start with first packet.
: 698      0954 2  !-
: 699      0955 2  CURR_PP=.PBCB[PBCB_A_PP_NEXT];
: 700      0956 2  WHILE .CURR_PP NEQ .PBCB[PBCB_A_PP_NEXT] DO
: 701      0957 2  BEGIN ! Update a pasting packet
: 702      0958 2  LOCAL
: 703      0959 2  PP_BASE : REF $PP_DECL; ! Base addr of this PP
: 704      0960 2  PP_BASE = .CURR_PP - PP_PBCB_QUEUE_OFFSET; ! Since queue header
: 705      0961 2  ! not at top of
: 706      0962 2  ! structure.
: 707      0963 2  STATUS=SMG$$CALC_PASTE_TRANSF(.PP_BASE);
: 708      0964 2  IF NOT .STATUS THEN RETURN .STATUS;
: 709      0965 2  CURR_PP = .PP_BASE [PP_A_NEXT_PBCB] ! Step to next PP
: 710      0966 2  END; ! Update a pasting packet
: 711      0967 2  !-
: 712      0968 2  !-
: 713      0969 2  !+
: 714      0970 2  ! Force an update.
: 715      0971 2  !-
: 716      0972 2  !-
: 717      0973 2  STATUS=SMG$$CHECK_FOR_OUTPUT_PBCB(.PBCB);
: 718      0974 2  IF NOT .STATUS THEN SIGNAL(.STATUS);
: 719      0975 2  !-
: 720      0976 2  END; ! Update all pasting packets
: 721      0977 2  !-
: 722      0978 2  !+
: 723      0979 2  ! If a new background color is desired, go do that now.
: 724      0980 2  !-
: 725      0981 2  !-
: 726      0982 2  IF NOT NULLPARAMETER(P_DESIRED_BACKGROUND_COLOR)
: 727      0983 2  THEN BEGIN ! Change background color
: 728      0984 2  BIND DESIRED_COLOR=.P_DESIRED_BACKGROUND_COLOR;
: 729      0985 2  BIND RESULTING_COLOR=PBCB[PBCB_B_BACKGROUND_COLOR];
: 730      0986 2  !-

```



```

731 0987 3 IF .DESIRED_COLOR EQL SMG$C_COLOR WHITE
732 0988 4 THEN $SMG$GET_TERM_DATA(LIGHT_SCREEN)
733 0989 3 ELSE $SMG$GET_TERM_DATA(DARK_SCREEN);
734 0990 3
735 0991 3 IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
736 0992 4 THEN BEGIN
737 0993 4 STATUS = SMG$OUTPUT(.PBCB,.PBCB[PBCB_L_CAP_LENGTH],
738 0994 4 .PBCB[PBCB_A_CAP_BUFFER]);
739 0995 4 IF NOT .STATUS THEN RETURN .STATUS
740 0996 3 END;
741 0997 3
742 0998 3 RESULTING_COLOR=.DESIRED_COLOR;
743 0999 3
744 1000 2 END; ! Change terminal color
745 1001 2
746 1002 2 !+
747 1003 2 ! If the user wants the new background color, give it to him now.
748 1004 2 !-
749 1005 2
750 1006 2 IF NOT NULLPARAMETER(P_RESULTING_BACKGROUND_COLOR)
751 1007 3 THEN BEGIN ! Return background color
752 1008 3 BIND RESULTING_COLOR=.P_RESULTING_BACKGROUND_COLOR;
753 1009 3 RESULTING_COLOR=.PBCB[PBCB_B_BACKGROUND_COLOR]
754 1010 2 END; ! Return background color
755 1011 2
756 1012 2 RETURN SS$_NORMAL
757 1013 2
758 1014 1 END; ! Routine SMG$CHANGE_PBD_CHARACTERISTICS

```

.TITLE SMG\$DISPLAY\_LINKS SMG\$DISPLAY\_LINKS - Virtual Display Linkages

.IDENT \1-096\

.PSECT \_SMG\$DATA,NOEXE, PIC,2

```

00000000 00000 PBD_L_COUNT::
               .LONG 0
00000000# 00004 PBD_A_PBCB::
               .LONG 0[16]
00# 00044 PBD_V_PB_AVAIL::
               .BYTE 0[2]
00000000 00046 .BLKB 2
00000000 00048 ZERO: .LONG 0
00000010 0004C PBD_K_MAX_PB_BY_REF:
               .LONG -16

```

```

.EXTRN LIB$ANALYZE_SDESC R2
.EXTRN LIB$FREE_VM, LIB$FREE_EF
.EXTRN LIB$GET_EF, LIB$GET_VM
.EXTRN LIB$COPY_DXDX, LIB$FREE1_DD
.EXTRN SMG$BEGIN_PASTEBOARD_UPDATE R1
.EXTRN SMG$END_PASTEBOARD_UPDATE_R2
.EXTRN SMG$ERASE_PASTEBOARD
.EXTRN SMG$CHECK_FOR_OUTPUT_DCB
.EXTRN SMG$CHECK_FOR_OUTPUT_PBCB
.EXTRN SMG$FILL_WINDOW_BUFFER

```



PC	OP	OP2	OP3	OP4	OP5	OP6	OP7	OP8	OP9	OP10	OP11	OP12	OP13	OP14	OP15	OP16	OP17	OP18	OP19	OP20	OP21	OP22	OP23	OP24	OP25	OP26	OP27	OP28	OP29	OP30	OP31	OP32	OP33	OP34	OP35	OP36	OP37	OP38	OP39	OP40	OP41	OP42	OP43	OP44	OP45	OP46	OP47	OP48	OP49	OP50	OP51	OP52	OP53	OP54	OP55	OP56	OP57	OP58	OP59	OP60	OP61	OP62	OP63	OP64	OP65	OP66	OP67	OP68	OP69	OP70	OP71	OP72	OP73	OP74	OP75	OP76	OP77	OP78	OP79	OP80	OP81	OP82	OP83	OP84	OP85	OP86	OP87	OP88	OP89	OP90	OP91	OP92	OP93	OP94	OP95	OP96	OP97	OP98	OP99	OP100	OP101	OP102	OP103	OP104	OP105	OP106	OP107	OP108	OP109	OP110	OP111	OP112	OP113	OP114	OP115	OP116	OP117	OP118	OP119	OP120	OP121	OP122	OP123	OP124	OP125	OP126	OP127	OP128	OP129	OP130	OP131	OP132	OP133	OP134	OP135	OP136	OP137	OP138	OP139	OP140	OP141	OP142	OP143	OP144	OP145	OP146	OP147	OP148	OP149	OP150	OP151	OP152	OP153	OP154	OP155	OP156	OP157	OP158	OP159	OP160	OP161	OP162	OP163	OP164	OP165	OP166	OP167	OP168	OP169	OP170	OP171	OP172	OP173	OP174	OP175	OP176	OP177	OP178	OP179	OP180	OP181	OP182	OP183	OP184	OP185	OP186	OP187	OP188	OP189	OP190	OP191	OP192	OP193	OP194	OP195	OP196	OP197	OP198	OP199	OP200	OP201	OP202	OP203	OP204	OP205	OP206	OP207	OP208	OP209	OP210	OP211	OP212	OP213	OP214	OP215	OP216	OP217	OP218	OP219	OP220	OP221	OP222	OP223	OP224	OP225	OP226	OP227	OP228	OP229	OP230	OP231	OP232	OP233	OP234	OP235	OP236	OP237	OP238	OP239	OP240	OP241	OP242	OP243	OP244	OP245	OP246	OP247	OP248	OP249	OP250	OP251	OP252	OP253	OP254	OP255	OP256	OP257	OP258	OP259	OP260	OP261	OP262	OP263	OP264	OP265	OP266	OP267	OP268	OP269	OP270	OP271	OP272	OP273	OP274	OP275	OP276	OP277	OP278	OP279	OP280	OP281	OP282	OP283	OP284	OP285	OP286	OP287	OP288	OP289	OP290	OP291	OP292	OP293	OP294	OP295	OP296	OP297	OP298	OP299	OP300	OP301	OP302	OP303	OP304	OP305	OP306	OP307	OP308	OP309	OP310	OP311	OP312	OP313	OP314	OP315	OP316	OP317	OP318	OP319	OP320	OP321	OP322	OP323	OP324	OP325	OP326	OP327	OP328	OP329	OP330	OP331	OP332	OP333	OP334	OP335	OP336	OP337	OP338	OP339	OP340	OP341	OP342	OP343	OP344	OP345	OP346	OP347	OP348	OP349	OP350	OP351	OP352	OP353	OP354	OP355	OP356	OP357	OP358	OP359	OP360	OP361	OP362	OP363	OP364	OP365	OP366	OP367	OP368	OP369	OP370	OP371	OP372	OP373	OP374	OP375	OP376	OP377	OP378	OP379	OP380	OP381	OP382	OP383	OP384	OP385	OP386	OP387	OP388	OP389	OP390	OP391	OP392	OP393	OP394	OP395	OP396	OP397	OP398	OP399	OP400	OP401	OP402	OP403	OP404	OP405	OP406	OP407	OP408	OP409	OP410	OP411	OP412	OP413	OP414	OP415	OP416	OP417	OP418	OP419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------



		21	11	00081		BRB	9\$		
		08	AE	D4	00083	8\$: CLRL	INPUT_ARGS		
		08	AE	9F	00086	PUSHAB	INPUT_ARGS		
		0104	C4	DD	00089	PUSHL	260(PBCB)		
			53	DD	0008D	PUSHL	R3		
		0100	C4	9F	0008F	PUSHAB	256(PBCB)		
14	AE	0104	8F	3C	00093	MOVZWL	#474, 20(SP)		
		14	AE	9F	00099	PUSHAB	20(SP)		
			52	DD	0009C	PUSHL	R2		
	6A		06	FB	0009E	CALLS	#6, SMG\$GET_TERM_DATA		
	3A		50	E9	000A1	BLBC	STATUS, 12\$		
			63	D5	000A4	9\$: TSTL	(R3)		0773
			11	13	000A6	BEQL	10\$		
		0104	C4	DD	000A8	PUSHL	260(PBCB)		0776
			63	DD	000AC	PUSHL	(R3)		0775
			54	DD	000AE	PUSHL	PBCB		
	6B		03	FB	000B0	CALLS	#3, SMG\$\$OUTPUT		
	55		50	D0	000B3	MOVL	R0, STATUS		
	3F		55	E9	000B6	BLBC	STATUS, 14\$		0777
			62	D5	000B9	10\$: TSTL	(R2)		0784
			04	12	000BB	BNEQ	11\$		
			63	D4	000BD	CLRL	(R3)		
			20	11	000BF	BRB	13\$		
		08	AE	D4	000C1	11\$: CLRL	INPUT_ARGS		
		08	AE	9F	000C4	PUSHAB	INPUT_ARGS		
		0104	C4	DD	000C7	PUSHL	260(PBCB)		
			53	DD	000CB	PUSHL	R3		
		0100	C4	9F	000CD	PUSHAB	256(PBCB)		
14	AE	DD	8F	9A	000D1	MOVZBL	#221, 20(SP)		
		14	AE	9F	000D6	PUSHAB	20(SP)		
			52	DD	000D9	PUSHL	R2		
	6A		06	FB	000DB	CALLS	#6, SMG\$GET_TERM_DATA		
	49		50	E9	000DE	12\$: BLBC	STATUS, 18\$		
			63	D5	000E1	13\$: TSTL	(R3)		0785
			1B	13	000E3	BEQL	15\$		
	56	0104	C4	D0	000E5	MOVL	260(PBCB), R6		0787
		0104	C4	DD	000EA	PUSHL	260(PBCB)		0789
			63	DD	000EE	PUSHL	(R3)		0788
			54	DD	000F0	PUSHL	PBCB		
	6B		03	FB	000F2	CALLS	#3, SMG\$\$OUTPUT		
	55		50	D0	000F5	MOVL	R0, STATUS		
	49		55	E9	000F8	14\$: BLBC	STATUS, 20\$		0790
	57		66	D0	000FB	MOVL	(R6), NORMAL_WIDTH		0791
			04	11	000FE	BRB	16\$		
	57	50	8F	9A	00100	15\$: MOVZBL	#80, NORMAL_WIDTH		0793
			62	D5	00104	16\$: TSTL	(R2)		0799
			04	12	00106	BNEQ	17\$		
			63	D4	00108	CLRL	(R3)		
			21	11	0010A	BRB	19\$		
		08	AE	D4	0010C	17\$: CLRL	INPUT_ARGS		
		08	AE	9F	0010F	PUSHAB	INPUT_ARGS		
		0104	C4	DD	00112	PUSHL	260(PBCB)		
			53	DD	00116	PUSHL	R3		
		0100	C4	9F	00118	PUSHAB	256(PBCB)		
14	AE	0246	8F	3C	0011C	MOVZWL	#582, 20(SP)		
		14	AE	9F	00122	PUSHAB	20(SP)		
			52	DD	00125	PUSHL	R2		



6A		06	FB	00127	CALLS	#6, SMG\$GET_TERM_DATA	:
7C		50	E9	0012A	BLBC	STATUS, 27\$	:
		63	D5	0012D	TSTL	(R3)	0800
		1B	13	0012F	BEQL	21\$	:
56	0104	C4	D0	00131	MOVL	260(PBCB), R6	0802
	0104	C4	DD	00136	PUSHL	260(PBCB)	0804
		63	DD	0013A	PUSHL	(R3)	0803
		54	DD	0013C	PUSHL	PBCB	:
6B		03	FB	0013E	CALLS	#3, SMG\$\$OUTPUT	:
55		50	D0	00141	MOVL	R0, STATUS	:
7B		55	E9	00144	BLBC	STATUS, 30\$	0805
56		66	D0	00147	MOVL	(R6), WIDE_WIDTH	0806
		04	11	0014A	BRB	22\$	:
56	50	8F	9A	0014C	MOVZBL	#80, WIDE_WIDTH	0808
57		58	D1	00150	CMPL	R8, NORMAL_WIDTH	0814
		2E	14	00153	BGTR	25\$	:
		62	D5	00155	TSTL	(R2)	0816
		04	12	00157	BNEQ	23\$	:
		63	D4	00159	CLRL	(R3)	:
		21	11	0015B	BRB	24\$	:
	08	AE	D4	0015D	CLRL	INPUT_ARGS	:
	08	AE	9F	00160	PUSHAB	INPUT_ARGS	:
	0104	C4	DD	00163	PUSHL	260(PBCB)	:
		53	DD	00167	PUSHL	R3	:
	0100	C4	9F	00169	PUSHAB	256(PBCB)	:
14	AE	0245	8F	3C	MOVZWL	#581, 20(SP)	:
	14	AE	9F	00173	PUSHAB	20(SP)	:
		52	DD	00176	PUSHL	R2	:
6A		06	FB	00178	CALLS	#6, SMG\$GET_TERM_DATA	:
2B		50	E9	0017B	BLBC	STATUS, 27\$	:
52		57	D0	0017E	MOVL	NORMAL_WIDTH, RESULTANT_WIDTH	0817
		2D	11	00181	BRB	29\$	:
		62	D5	00183	TSTL	(R2)	0820
		04	12	00185	BNEQ	26\$	:
		63	D4	00187	CLRL	(R3)	:
		22	11	00189	BRB	28\$	:
	08	AE	D4	0018B	CLRL	INPUT_ARGS	:
	08	AE	9F	0018E	PUSHAB	INPUT_ARGS	:
	0104	C4	DD	00191	PUSHL	260(PBCB)	:
		53	DD	00195	PUSHL	R3	:
	0100	C4	9F	00197	PUSHAB	256(PBCB)	:
14	AE	0246	8F	3C	MOVZWL	#582, 20(SP)	:
	14	AE	9F	001A1	PUSHAB	20(SP)	:
		52	DD	001A4	PUSHL	R2	:
6A		06	FB	001A6	CALLS	#6, SMG\$GET_TERM_DATA	:
01		50	E8	001A9	BLBS	STATUS, 28\$	:
		04	001AC	RET			:
52		56	D0	001AD	MOVL	WIDE_WIDTH, RESULTANT_WIDTH	0821
		63	D5	001B0	TSTL	(R3)	0824
		11	13	001B2	BEQL	31\$	:
	0104	C4	DD	001B4	PUSHL	260(PBCB)	0827
		63	DD	001B8	PUSHL	(R3)	0826
		54	DD	001BA	PUSHL	PBCB	:
6B		03	FB	001BC	CALLS	#3, SMG\$\$OUTPUT	:
55		50	D0	001BF	MOVL	R0, STATUS	:
72		55	E9	001C2	BLBC	STATUS, 39\$	0828
50		52	D0	001C5	MOVL	RESULTANT_WIDTH, R0	0837



			58		50	D1	001C8		CMPL	R0, R8		
					03	1B	001CB		BLEQU	32\$		
			50		58	D0	001CD		MOVL	R8, R0		
			52		50	D0	001D0	32\$:	MOVL	R0, RESULTANT_WIDTH		
			50	5A	A4	3C	001D3		MOVZWL	90(PBCB), PREVIOUS_WIDTH		0853
			A4		52	B0	001D7		MOVW	RESULTANT_WIDTH, 90(PBCB)		0854
			59		01	D0	001DB		MOVL	#1, PASTING_PACKET_PANIC		0863
			03		6C	91	001DE	33\$:	CMPB	(AP), #3		0878
					0A	1F	001E1		BLSSU	34\$		
				0C	AC	D5	001E3		TSTL	12(AP)		
					05	13	001E6		BEQL	34\$		
			0C	BC	5A	A4	3C	001E8	MOVZWL	90(PBCB), @P_RESULTING_WIDTH		0881
			04		6C	91	001ED	34\$:	CMPB	(AP), #4		0892
					4F	1F	001F0		BLSSU	41\$		
				10	AC	D5	001F2		TSTL	16(AP)		
					4A	13	001F5		BEQL	41\$		
				00A4	C4	D5	001F7		TSTL	164(PBCB)		0895
					08	13	001FB		BEQL	36\$		
			50	00000000G	8F	D0	001FD	35\$:	MOVL	#SMG\$_PBDIN_USE, R0		0896
					04	00204			RET			
			52	10	BC	D0	00205	36\$:	MOVL	@P_DESIRED_HEIGHT, R2		0897
					08	12	00209		BNEQ	37\$		
			50	00000000G	8F	D0	0020B		MOVL	#SMG\$_INVPAGARG, R0		0898
					04	00212			RET			
52	5F	A4	08		00	ED	00213	37\$:	CMPZV	#0, #8, 95(PBCB), R2		0899
					26	13	00219		BEQL	41\$		
			18		52	D1	0021B		CMPL	R2, #24		0906
					03	1B	0021E		BLEQU	38\$		
			52		18	D0	00220		MOVL	#24, R2		
52	5F	A4	08		00	ED	00223	38\$:	CMPZV	#0, #8, 95(PBCB), R2		
					0F	1B	00229		BLEQU	40\$		
					54	DD	0022B		PUSHL	PBCB		0908
				00000000G	01	FB	0022D		CALLS	#1, SMG\$\$ERASE_PASTEBOARD		
					50	D0	00234		MOVL	R0, STATUS		
					55	E9	00237	39\$:	BLBC	STATUS, 44\$		0909
			5F	A4	52	90	0023A	40\$:	MOVB	R2, 95(PBCB)		0914
					01	D0	0023E		MOVL	#1, PASTING_PACKET_PANIC		0915
			59		6C	91	00241	41\$:	CMPB	(AP), #5		0923
			05		0A	1F	00244		BLSSU	42\$		
				14	AC	D5	00246		TSTL	20(AP)		
					05	13	00249		BEQL	42\$		
			14	BC	5F	A4	9A	0024B	MOVZBL	95(PBCB), @P_RESULTING_HEIGHT		0926
				6C	59	E9	00250	42\$:	BLBC	PASTING_PACKET_PANIC, 47\$		0935
					08	A4	DD	00253	PUSHL	8(PBCB)		0941
				00000000G	01	FB	00256		CALLS	#1, SMG\$\$DEALLOCATE_WCB		
					50	D0	0025D		MOVL	R0, STATUS		
					55	E9	00260		BLBC	STATUS, 44\$		0942
					08	A4	9F	00263	PUSHAB	8(PBCB)		0948
			08	AE	5A	A4	3C	00266	MOVZWL	90(PBCB), 8(SP)		0947
					08	AE	9F	0026B	PUSHAB	8(SP)		
			08	AE	5F	A4	9A	0026E	MOVZBL	95(PBCB), 8(SP)		0946
					08	AE	9F	00273	PUSHAB	8(SP)		
				00000000G	00	03	FB	00276	CALLS	#3, SMG\$\$CREATE_WCB		0948
					55	D0	0027D		MOVL	R0, STATUS		
					18	E9	00280		BLBC	STATUS, 44\$		0949
					53	D0	00283		MOVL	(PBCB), CURR_PP		0955
			54		53	D1	00286	43\$:	CMPL	CURR_PP, PBCB		0956



	52	F8	1C	13	00289	BEQL	46\$		
			A3	9E	0028B	MOVAB	-8(R3), PP_BASE		0961
			52	DD	0028F	PUSHL	PP_BASE		0964
00000000G	00		01	FB	00291	CALLS	#1, SMG\$\$CALC_PASTE_TRANSF		
	55		50	D0	00298	MOVL	R0, STATUS		
	03		55	E8	0029B	BLBS	STATUS, 45\$		0965
			0092	31	0029E	BRW	53\$		
	53	08	A2	D0	002A1	MOVL	8(PP_BASE), CURR_PP		0966
			DF	11	002A5	BRB	43\$		
			54	DD	002A7	PUSHL	PBCB		0973
00000000G	00		01	FB	002A9	CALLS	#1, SMG\$\$CHECK_FOR_OUTPUT_PBCB		
	55		50	D0	002B0	MOVL	R0, STATUS		
	09		55	E8	002B3	BLBS	STATUS, 47\$		0974
			55	DD	002B6	PUSHL	STATUS		
00000000G	00		01	FB	002B8	CALLS	#1, LIB\$SIGNAL		
	06		6C	91	002BF	CMPB	(AP), #6		0982
			79	1F	002C2	BLSSU	55\$		
		18	AC	D5	002C4	TSTL	24(AP)		
			74	13	002C7	BEQL	55\$		
	50	00FC	C4	9E	002C9	MOVAB	252(PBCB), R0		0988
	52	0108	C4	9E	002CE	MOVAB	264(PBCB), R2		
	01	18	BC	D1	002D3	CMPL	@P_DESIRED_BACKGROUND_COLOR, #1		0987
			1C	12	002D7	BNEQ	48\$		
			60	D5	002D9	TSTL	(R0)		0988
			1C	13	002DB	BEQL	49\$		
		08	AE	D4	002DD	CLRL	INPUT_ARGS		
		08	AE	9F	002E0	PUSHAB	INPUT_ARGS		
		0104	C4	DD	002E3	PUSHL	260(PBCB)		
			52	DD	002E7	PUSHL	R2		
		0100	C4	9F	002E9	PUSHAB	256(PBCB)		
14	AE	0228	8F	3C	002ED	MOVZWL	#552, 20(SP)		
			1E	11	002F3	BRB	51\$		
			60	D5	002F5	TSTL	(R0)		0989
			04	12	002F7	BNEQ	50\$		
			62	D4	002F9	CLRL	(R2)		
			21	11	002FB	BRB	52\$		
		08	AE	D4	002FD	CLRL	INPUT_ARGS		
		08	AE	9F	00300	PUSHAB	INPUT_ARGS		
		0104	C4	DD	00303	PUSHL	260(PBCB)		
			52	DD	00307	PUSHL	R2		
		0100	C4	9F	00309	PUSHAB	256(PBCB)		
14	AE	01C8	8F	3C	0030D	MOVZWL	#456, 20(SP)		
		14	AE	9F	00313	PUSHAB	20(SP)		
			50	DD	00316	PUSHL	R0		
	6A		06	FB	00318	CALLS	#6, SMG\$GET_TERM_DATA		
	32		50	E9	0031B	BLBC	STATUS, 57\$		0991
			62	D5	0031E	TSTL	(R2)		
			15	13	00320	BEQL	54\$		
		0104	C4	DD	00322	PUSHL	260(PBCB)		0994
			62	DD	00326	PUSHL	(R2)		0993
			54	DD	00328	PUSHL	PBCB		
	6B		03	FB	0032A	CALLS	#3, SMG\$\$OUTPUT		
	55		50	D0	0032D	MOVL	R0, STATUS		
	04		55	E8	00330	BLBS	STATUS, 54\$		0995
	50		55	D0	00333	MOVL	STATUS, R0		
			04	00336	RET				
00F9	C4	18	BC	D0	00337	MOVL	@P_DESIRED_BACKGROUND_COLOR, 249(PBCB)		0998



SMG\$DISPLAY_LIN	SMG\$DISPLAY_LINKS - Virtual Display Linkages	G 13	16-Sep-1984 00:29:22	VAX-11 Bliss-32 V4.0-742	Page 23
1-096	SMG\$CHANGE_PBD_CHARACTERISTICS		14-Sep-1984 13:09:43	[SMGRTL.SRC]SMGDISLIN.B32;1	(6)

07	6C	91	0033D	55\$:	CMPB	(AP), #7	:	1006
	0B	1F	00340		BLSSU	56\$	:	
	1C	AC	D5	00342	TSTL	28(AP)	:	
	06	13	00345		BEQL	56\$	:	
1C	BC	00F9	C4	9A	00347	MOVZBL	249(PBCB), @P_RESULTING_BACKGROUND_COLOR	1009
	50		01	D0	0034D	MOVL	#1, R0	1012
			04	00350	57\$:	RET	:	1014

; Routine Size: 849 bytes,      Routine Base: \_SMG\$CODE + 0000



```

760 1015 1 %SBTTL 'SMG$CHANGE_VIRTUAL_DISPLAY - Change Virtual Display'
761 1016 1 GLOBAL ROUTINE SMG$CHANGE_VIRTUAL_DISPLAY (
762 1017 1     DISPLAY_ID,
763 1018 1     NUM_ROWS,
764 1019 1     NUM_COLS,
765 1020 1     DISPLAY_ATTRIBUTES,
766 1021 1     VIDEO_ATTRIBUTES,
767 1022 1     CHAR_SET
768 1023 1 ) =
769 1024 1
770 1025 1 ++
771 1026 1 FUNCTIONAL DESCRIPTION:
772 1027 1     This routine changes the size or default attributes of an
773 1028 1     existing virtual display. The text which is currently in this
774 1029 1     virtual display is remapped to fit the new dimensions starting
775 1030 1     at row 1 column 1. Resulting cursor position will be at row 1
776 1031 1     column 1.
777 1032 1
778 1033 1 CALLING SEQUENCE:
779 1034 1
780 1035 1     ret_status.wlc.v = SMG$CHANGE_VIRTUAL_DISPLAY (
781 1036 1         DISPLAY_ID.rl.r,
782 1037 1         [,NUM_ROWS.rl.r]
783 1038 1         [,NUM_COLS.rl.r]
784 1039 1         [,DISPLAY_ATTRIBUTES.rl.r]
785 1040 1         [,VIDEO_ATTRIBUTES.rl.r]
786 1041 1         [,CHAR_SET.rl.r])
787 1042 1
788 1043 1 FORMAL PARAMETERS:
789 1044 1
790 1045 1     DISPLAY_ID.rl.r Display id of virtual display to be changed.
791 1046 1
792 1047 1     NUM_ROWS.rl.r  Number of rows in new virtual display.
793 1048 1     If omitted, the number of rows remains the same.
794 1049 1
795 1050 1     NUM_COLS.rl.r  Number of columns in new virtual display.
796 1051 1     If omitted, the number of columns remains the same.
797 1052 1
798 1053 1     DISPLAY_ATTRIBUTES.rl.r The default display attributes:
799 1054 1
800 1055 1         SMG$M_BORDER if virtual display is to be
801 1056 1         displayed with a border.
802 1057 1
803 1058 1         SMG$M_TRUNC_ICON if an icon should be displayed
804 1059 1         when text overflows the display bounds.
805 1060 1
806 1061 1         SMG$M_DISPLAY_CONTROLS if carriage controls (CR, LF,
807 1062 1         TFF, VT, HT) should be displayed instead
808 1063 1         of executed.
809 1064 1
810 1065 1         If omitted, the default display attributes
811 1066 1         currently associated with the display will be
812 1067 1         retained.
813 1068 1
814 1069 1     VIDEO_ATTRIBUTES.rl.r The default rendition code to be
815 1070 1     applied to all output to this display unless
816 1071 1     overridden on a particular output call.
  
```



```

: 817      1072 1 |
: 818      1073 1 |
: 819      1074 1 |
: 820      1075 1 |
: 821      1076 1 |
: 822      1077 1 |
: 823      1078 1 |
: 824      1079 1 |
: 825      1080 1 |
: 826      1081 1 |
: 827      1082 1 |
: 828      1083 1 |
: 829      1084 1 |
: 830      1085 1 |
: 831      1086 1 |
: 832      1087 1 |
: 833      1088 1 |
: 834      1089 1 |
: 835      1090 1 |
: 836      1091 1 |
: 837      1092 1 |
: 838      1093 1 |
: 839      1094 1 |
: 840      1095 1 |
: 841      1096 1 |
: 842      1097 1 |
: 843      1098 1 |
: 844      1099 1 |
: 845      1100 1 |
: 846      1101 1 |
: 847      1102 1 |
: 848      1103 1 |
: 849      1104 1 |
: 850      1105 1 |
: 851      1106 1 |
: 852      1107 1 |
: 853      1108 1 |
: 854      1109 1 |
: 855      1110 1 |
: 856      1111 1 |
: 857      1112 1 |
: 858      1113 1 |
: 859      1114 1 |
: 860      1115 1 |
: 861      1116 1 |
: 862      1117 1 |
: 863      1118 1 |
: 864      1119 1 |
: 865      1120 1 |
: 866      1121 1 |
: 867      1122 2 |
: 868      1123 2 |
: 869      1124 2 |
: 870      1125 2 |
: 871      1126 2 |
: 872      1127 2 |
: 873      1128 2 |

      If omitted, the current video attributes are
      retained.

      Values:

      SMG$M_BLINK      displays characters blinking.

      SMG$M_BOLD      displays characters in
                      higher-than-normal intensity.

      SMG$M_REVERSE    displays characters in reverse
                      video -- that is, using the
                      opposite default rendition of
                      the virtual display.

      SMG$M_UNDERLINE  displays characters underlined.

      CHAR_SET.rl.r    The default character set for all text
                      associated with this display.
                      Recognized values are:
                      SMG$C_UNITED_KINGDOM
                      SMG$C_ASCII (default)
                      SMG$C_SPEC_GRAPHICS
                      SMG$C_ALT_CHAR
                      SMG$C_ALT_GRAPHICS

      IMPLICIT INPUTS:

      NONE

      IMPLICIT OUTPUTS:

      NONE

      COMPLETION STATUS:

      SSS$ NORMAL      Normal successful completion
      LIB$_INSVIRMEM    Insufficient virtual memory to reallocate needed
                      buffers.
      SMG$_INVARG       Unrecognized Video Attributes
                      or Unrecognized Display Attributes
      SMG$_WRONUMARG    Wrong number of arguments.

      SIDE EFFECTS:

      Cursor for virtual display will be forced to row 1 column 1 if
      display is redimensioned.
      If a labeled border applies and does not fit newly redimensioned
      display, the label will be deleted.

      --
      BEGIN
      BUILTIN
      NULLPARAMETER;

      LOCAL
      STATUS,           ! Status of subroutine calls
      PP : REF $PP_DECL, ! Addr. of a pasting packet

```



```

874      1129 2      NEW_ROWS,      ! New number of rows
875      1130 2      NEW_COLS,      ! New number of columns
876      1131 2      DCB: REF $DCB_DECL, ! Addr of display control block
877      1132 2      NEW_SIZE;      ! New rows * columns
878      1133 2
879      1134 2      $SMG$VALIDATE_ARGCOUNT (1, 6);      ! Test for right no. of args
880      1135 2
881      1136 2      $SMG$GET_DCB (.DISPLAY_ID, DCB);      ! Get address of virtual display
882      1137 2      ! control block.
883      1138 2
884      1139 2      !+
885      1140 2      ! Determine size of new buffer we need.
886      1141 2      !-
887      1142 2      IF NOT NULLPARAMETER (NUM_ROWS)      ! If new number of rows specified
888      1143 2      THEN
889      1144 2      NEW_ROWS = ..NUM_ROWS
890      1145 2      ELSE
891      1146 2      NEW_ROWS = .DCB [DCB_W_NO_ROWS];
892      1147 2
893      1148 2      IF NOT NULLPARAMETER (NUM_COLS)      ! If new number of columns specified
894      1149 2      THEN
895      1150 2      NEW_COLS = ..NUM_COLS
896      1151 2      ELSE
897      1152 2      NEW_COLS = .DCB [DCB_W_NO_COLS];
898      1153 2
899      1154 2      NEW_SIZE = .NEW_ROWS * .NEW_COLS;
900      1155 2
901      1156 2      !+
902      1157 2      ! Adjust default display, video attributes and default character set if
903      1158 2      ! they are specified.
904      1159 2      !-
905      1160 2      IF NOT NULLPARAMETER (DISPLAY_ATTRIBUTES) ! If display attributes specified
906      1161 2      THEN
907      1162 2      DCB [DCB_B_DEF_DISPLAY_ATTR] = ..DISPLAY_ATTRIBUTES;
908      1163 2
909      1164 2      IF NOT NULLPARAMETER (VIDEO_ATTRIBUTES) ! If video attributes specified
910      1165 2      THEN
911      1166 2      DCB [DCB_B_DEF_VIDEO_ATTR] = ..VIDEO_ATTRIBUTES;
912      1167 2
913      1168 2      IF NOT NULLPARAMETER (CHAR_SET) ! If char set specified
914      1169 2      THEN
915      1170 2      DCB [DCB_B_DEF_CHAR_SET] = ..CHAR_SET;
916      1171 2
917      1172 2      !+
918      1173 2      ! If the dimensions of the old buffer and the new buffers are different,
919      1174 2      ! we will have to allocate new buffer space and copy existing text into
920      1175 2      ! new buffers.
921      1176 2      !-
922      1177 2      IF .DCB [DCB_L_BUFSIZE] NEQ .NEW_SIZE      OR
923      1178 2      .DCB [DCB_W_NO_ROWS] NEQ .NEW_ROWS      OR
924      1179 2      .DCB [DCB_W_NO_COLS] NEQ .NEW_COLS
925      1180 2      THEN
926      1181 3      BEGIN ! Redimensioning required
927      1182 3      LOCAL
928      1183 3      STATUS, ! Status of subroutine calls
929      1184 3      ROWS_TO_MOVE, ! No of rows that will be moved from
930      1185 3      ! old buffer to new.

```



```

: 931      1186 3      COLS_TO_MOVE,      ! No of columns that will be moved from
: 932      1187 3      ! old to new.
: 933      1188 3      NEW_TEXT_BUF : REF VECTOR [,BYTE],      ! Addr of new text
: 934      1189 3      ! buffer
: 935      1190 3      NEW_ATTR_BUF : REF VECTOR [,BYTE],      ! Addr of new attr
: 936      1191 3      ! buffer
: 937      1192 3      NEW_CHAR_BUF : REF VECTOR [,BYTE],      ! Addr of new char_set
: 938      1193 3      ! buffer
: 939      1194 3
: 940      1195 3      TEXT_PTR : REF VECTOR [, BYTE],      ! Address of current
: 941      1196 3      ! text buffer in DCB.
: 942      1197 3
: 943      1198 3      ATTR_PTR : REF VECTOR [,BYTE],      ! Address of current
: 944      1199 3      ! attr buffer in DCB
: 945      1200 3
: 946      1201 3      CHAR_PTR : REF VECTOR [,BYTE];      ! Address of current
: 947      1202 3      ! char_set buffer in
: 948      1203 3      ! DCB
: 949      1204 3
: 950      1205 3      !+
: 951      1206 3      ! Get space for two new, properly-dimensioned buffers.
: 952      1207 3      !-
: 953      1208 4      IF NOT (STATUS = LIB$GET_VM (%REF (2 * .NEW_SIZE),
: 954      1209 4      NEW_TEXT_BUF))
: 955      1210 3      THEN
: 956      1211 3      RETURN (.STATUS);
: 957      1212 3
: 958      1213 3      NEW_ATTR_BUF = .NEW_TEXT_BUF + .NEW_SIZE;
: 959      1214 3
: 960      1215 3      !+
: 961      1216 3      ! Now need to copy text and attribute information from
: 962      1217 3      ! .DCB [DCB_A_TEXT_BUF] and .DCB [DCB_A_ATTR_BUF] to
: 963      1218 3      ! .NEW_TEXT_BUF and .NEW_ATTR_BUF, preserving the line context.
: 964      1219 3      ! First pre-blank new text buffer and attribute buffer in
: 965      1220 3      ! case old do not cover new area.
: 966      1221 3      !-
: 967      1222 3      CH$FILL ( 'C' , .NEW_SIZE, .NEW_TEXT_BUF);
: 968      1223 3      CH$FILL ( .DCB [DCB_B_DEF_VIDEO_ATTR], .NEW_SIZE, .NEW_ATTR_BUF);
: 969      1224 3
: 970      1225 3      TEXT_PTR = .DCB [DCB_A_TEXT_BUF];
: 971      1226 3      ATTR_PTR = .DCB [DCB_A_ATTR_BUF];
: 972      1227 3      CHAR_PTR = .DCB [DCB_A_CHAR_SET_BUF];
: 973      1228 3
: 974      1229 3      ROWS_TO_MOVE = MIN (.DCB [DCB_W_NO_ROWS], .NEW_ROWS);
: 975      1230 3      COLS_TO_MOVE = MIN (.DCB [DCB_W_NO_COLS], .NEW_COLS);
: 976      1231 3
: 977      1232 3      INCR I FROM 1 TO .ROWS_TO_MOVE
: 978      1233 3      DO
: 979      1234 4      BEGIN      ! Move text and attrib. to new buffers.
: 980      1235 4      LOCAL
: 981      1236 4      SOURCE_INDEX,
: 982      1237 4      DEST_INDEX;
: 983      1238 4
: 984      1239 4      SOURCE_INDEX = (.I -1) * .DCB [DCB_W_NO_COLS] ;
: 985      1240 4      DEST_INDEX = (.I -1) * ..NUM_COLS ;
: 986      1241 4
: 987      1242 4      CH$MOVE ( .COLS_TO_MOVE,      ! No of chars.

```



```

: 988      1243  4      TEXT_PTR [.SOURCE_INDEX],      ! From
: 989      1244  4      NEW_TEXT_BUF [.DEST_INDEX]);    ! To
: 990      1245  4
: 991      1246  4      CH$MOVE ( .COLS_TO_MOVE,      ! No. of chars.
: 992      1247  4      ATTR_PTR [.SOURCE_INDEX],      ! From
: 993      1248  4      NEW_ATTR_BUF [.DEST_INDEX]);    ! To
: 994      1249  4
: 995      1250  4
: 996      1251  3      END;      ! Move text and attrib to new buffers.
: 997      1252  3
: 998      1253  3
: 999      1254  3      !+ Deal with alternate character set buffers if they exist.
1000      1255  3      !-
1001      1256  3      IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
1002      1257  3      THEN
1003      1258  4      BEGIN      ! Alt. char set buffer exists
1004      1259  4      !+
1005      1260  4      !- Allocate a new alternate character set buffer and init. it
1006      1261  4      !-
1007      1262  5      IF NOT (STATUS = LIB$GET_VM (NEW_SIZE, NEW_CHAR_BUF))
1008      1263  4      THEN
1009      1264  5      BEGIN
1010      1265  5      LIB$FREE_VM (%REF (2* .NEW_SIZE), NEW TEXT_BUF);
1011      1266  5      RETURN (.STATUS); ! Return LIB$_INSVIRMEM from GET call
1012      1267  4      END;
1013      1268  4
1014      1269  4      CH$FILL ( .DCB [DCB_B_DEF_CHAR_SET], .NEW_SIZE,
1015      1270  4      .NEW_CHAR_BUF);
1016      1271  4
1017      1272  4      !+
1018      1273  4      !- Move current contents row by row
1019      1274  4      !-
1020      1275  4      INCR I FROM 1 TO .ROWS_TO_MOVE
1021      1276  4      DO
1022      1277  5      BEGIN      ! Move loop
1023      1278  5      LOCAL
1024      1279  5      SOURCE_INDEX,
1025      1280  5      DEST_INDEX;
1026      1281  5
1027      1282  5      SOURCE_INDEX = (.I-1) * .DCB [DCB_W_NO_COLS] ;
1028      1283  5      DEST_INDEX = (.I-1) * ..NUM_COLS ;
1029      1284  5      CH$MOVE ( .COLS_TO_MOVE,      ! No of chars
1030      1285  5      CHAR_PTR [.SOURCE_INDEX],      ! From
1031      1286  5      NEW_CHAR_BUF [.DEST_INDEX]);    ! To
1032      1287  4      END;      ! Move loop
1033      1288  4
1034      1289  4      !+
1035      1290  4      !- Free old alternate char. set buffer and plug in new addr.
1036      1291  4      !-
1037      1292  5      IF NOT (STATUS = LIB$FREE_VM ( DCB [DCB_L_BUFSIZE],
1038      1293  5      DCB [DCB_A_CHAR_SET_BUF]))
1039      1294  4      THEN
1040      1295  4      RETURN (.STATUS);
1041      1296  4
1042      1297  4      DCB [DCB_A_CHAR_SET_BUF] = .NEW_CHAR_BUF;
1043      1298  4
1044      1299  3      END;      ! Alt. char set buffer exists

```



```

: 1045      1300      3
: 1046      1301      3
: 1047      1302      3
: 1048      1303      3
: 1049      1304      3
: 1050      1305      3
: 1051      1306      4
: 1052      1307      4
: 1053      1308      3
: 1054      1309      3
: 1055      1310      3
: 1056      1311      3
: 1057      1312      3
: 1058      1313      3
: 1059      1314      3
: 1060      1315      3
: 1061      1316      3
: 1062      1317      3
: 1063      1318      3
: 1064      1319      3
: 1065      1320      3
: 1066      1321      4
: 1067      1322      4
: 1068      1323      4
: 1069      1324      4
: 1070      1325      4
: 1071      1326      4
: 1072      1327      4
: 1073      1328      4
: 1074      1329      4
: 1075      1330      4
: 1076      1331      4
: 1077      1332      4
: 1078      1333      4
: 1079      1334      4
: 1080      1335      4
: 1081      1336      4
: 1082      1337      5
: 1083      1338      5
: 1084      1339      4
: 1085      1340      5
: 1086      1341      5
: 1087      1342      5
: 1088      1343      5
: 1089      1344      5
: 1090      1345      5
: 1091      1346      5
: 1092      1347      4
: 1093      1348      4
: 1094      1349      4
: 1095      1350      4
: 1096      1351      4
: 1097      1352      4
: 1098      1353      4
: 1099      1354      4
: 1100      1355      4
: 1101      1356      4

      !+
      ! Now that the text and attributes are safe in their
      ! new buffers, we release the old buffers and put the addresses
      ! of the new buffers in the DCB.
      !-
      IF NOT (STATUS = LIB$FREE_VM (%REF ( 2 * .DCB [DCB_L_BUFSIZE]),
                                   DCB [DCB_A_TEXT_BUF]))
      THEN
        RETURN (.STATUS);

      DCB [DCB_A_TEXT_BUF] = .NEW_TEXT_BUF;    ! Plug in new addresses
      DCB [DCB_A_ATTR_BUF] = .NEW_ATTR_BUF;

      !+
      ! If the number of rows changed, we need to reallocate the
      ! line characteristics vector and copy over as much of it as
      ! fits.
      !-
      IF .DCB [DCB_W_NO_ROWS] NEQ .NEW_ROWS
      THEN
        BEGIN      ! No. of rows changed
          LOCAL
            NEW_LINE_CHAR : REF VECTOR [,BYTE],      ! Addr of a new
                                                    ! line Char.
                                                    ! vector

            LINE_CHAR_PTR : REF VECTOR [,BYTE];      ! Addr of curr.
                                                    ! line Char.
                                                    ! vector

          LINE_CHAR_PTR = .DCB [DCB_A_LINE_CHAR];

          !+
          ! Allocate a new line characteristics vector of the right
          ! length. Quit if we can't get it.
          !-
          IF NOT (STATUS = LIB$GET_VM ( %REF (.NEW_ROWS + 1),
                                       NEW_LINE_CHAR))
          THEN
            BEGIN      ! Error path
              !+
              ! Give back all space acquired on this transaction,
              ! ignoring further errors, and quit.
              !-
              LIB$FREE_VM ( %REF (2 * .NEW_SIZE), NEW_TEXT_BUF);
              RETURN (.STATUS);
            END;      ! Error path

          !+
          ! Clear entire allocated vector to zero
          !-
          CH$FILL ( 0, .NEW_ROWS+1, .NEW_LINE_CHAR);

          !+
          ! Copy over as much of old line characteristics vector as
          ! will fit.

```



```

: 1102      1357  4      !-
: 1103      1358  4      CH$MOVE ( .ROWS TO MOVE,
: 1104      1359  4          LINE_CHAR_PTR [1],
: 1105      1360  4          NEW_LINE_CHAR [1]);
: 1106      1361  4
: 1107      1362  4
: 1108      1363  4      !+
: 1109      1364  4      !- Free former line characteristics vector.
: 1110      1365  5      IF NOT (STATUS = LIB$FREE_VM (
: 1111      1366  5          %REF (.DCB [DCB_W_NO_ROWS] +1),
: 1112      1367  5          DCB [DCB_A_LINE_CHAR]))
: 1113      1368  4      THEN
: 1114      1369  4          RETURN (.STATUS);
: 1115      1370  4
: 1116      1371  4      !+
: 1117      1372  4      !- Store address of new line characteristics vector in DCB
: 1118      1373  4
: 1119      1374  4      DCB [DCB_A_LINE_CHAR] = .NEW_LINE_CHAR;
: 1120      1375  3      END;          ! No. of rows changed
: 1121      1376  3
: 1122      1377  3      !+
: 1123      1378  3      !- Adjust the no. of rows and no. of cols. recorded in the DCB.
: 1124      1379  3
: 1125      1380  3      DCB [DCB_W_NO_ROWS] = .NEW_ROWS;          ! Adjust row/column size
: 1126      1381  3      DCB [DCB_W_NO_COLS] = .NEW_COLS;
: 1127      1382  3      DCB [DCB_L_BUFSIZE] = .NEW_SIZE;
: 1128      1383  3
: 1129      1384  3      !+
: 1130      1385  3      !- Force cursor to home.
: 1131      1386  3
: 1132      1387  3      DCB [DCB_W_CURSOR_ROW] = 1;
: 1133      1388  3      DCB [DCB_W_CURSOR_COL] = 1;
: 1134      1389  3
: 1135      1390  3      !+
: 1136      1391  3      !- Knock down flags that indicate we are at end of a row and that
: 1137      1392  3      !- we are in last line.
: 1138      1393  3
: 1139      1394  3      DCB [DCB_V_FULL] = 0;
: 1140      1395  3      DCB [DCB_V_COL_80] = 0;
: 1141      1396  3
: 1142      1397  3      !+
: 1143      1398  3      !- Reset the scrolling region within the redimensioned virtual
: 1144      1399  3      !- display to be the whole display.
: 1145      1400  3
: 1146      1401  3      DCB [DCB_W_TOP_OF_SCRREG] = 1;
: 1147      1402  3      DCB [DCB_W_BOTTOM_OF_SCRREG] = .NEW_ROWS;
: 1148      1403  3
: 1149      1404  3      !+
: 1150      1405  3      !- Now deal with border data, if any exists.
: 1151      1406  3
: 1152      1407  3      IF .DCB [DCB_V_BORDERED]
: 1153      1408  3      THEN
: 1154      1409  4          BEGIN          ! Bordered
: 1155      1410  4          LOCAL
: 1156      1411  4              DESC : REF BLOCK [8,BYTE]; ! Pointer to dynamic string
: 1157      1412  4              ! desc. for border label
: 1158      1413  4

```



```

: 1159      1414  4      DESC = DCB [DCB_Q_LABEL_DESC];
: 1160      1415  4      IF .DESC [DSC$A_POINTER] NEQ 0      ! If label exists
: 1161      1416  4      THEN
: 1162      1417  5          BEGIN      ! Label exists
: 1163      1418  5          LOCAL
: 1164      1419  5              TEMP;
: 1165      1420  5          TEMP = .DCB [DCB_W_LABEL_UNITS];
: 1166      1421  5
: 1167      1422  5          !+
: 1168      1423  5          ! Try to reapply our existing border label on this
: 1169      1424  5          ! redimensioned virtual display. If it now doesn't
: 1170      1425  5          ! fit because of the new dimensions, delete the label.
: 1171      1426  5          !-
: 1172      1427  6          IF NOT (SMG$LABEL BORDER (
: 1173      1428  6              .DISPLAY_ID,
: 1174      1429  6              .DESC,
: 1175      1430  6              %REF (.DCB [DCB_B_LABEL_POS]),
: 1176      1431  6              !+
: 1177      1432  6              ! Conditionalize UNITS parameter to
: 1178      1433  6              ! LABEL_BORDER depending on whether
: 1179      1434  6              ! caller originally specified
: 1180      1435  6              ! "centering" or gave us specific units.
: 1181      1436  6              !-
: 1182      1437  7              (IF .DCB [DCB_V_LABEL_CENTER] THEN 0
: 1183      1438  6                  ELSE TEMP),
: 1184      1439  6              %REF (.DCB [DCB_B_LABEL_REND])
: 1185      1440  6              ))
: 1186      1441  5          THEN
: 1187      1442  5              LIB$FREE1 DD (.DESC);      ! Delete label
: 1188      1443  4          END;      ! Label exists
: 1189      1444  3          END;      ! Bordered
: 1190      1445  2          END;      ! Redimensioning required
: 1191      1446  2
: 1192      1447  2      !+
: 1193      1448  2      ! Since the dimension of the virtual display may have changed, or we
: 1194      1449  2      ! may have added or deleted a border, we need to recalculate the
: 1195      1450  2      ! transformation constants that occur in each pasting packet we are
: 1196      1451  2      ! involved in.
: 1197      1452  2      ! Check to see if we can do it now or must wait because we are batched.
: 1198      1453  2      !-
: 1199      1454  2      IF .DCB [DCB_L_BATCH_LEVEL] EQL 0
: 1200      1455  2      THEN
: 1201      1456  3          BEGIN      ! Can do it now
: 1202      1457  3          LOCAL
: 1203      1458  3              CURR_PP : REF $PP_DECL;      ! Addr of a pasting packet
: 1204      1459  3
: 1205      1460  4          IF NOT (STATUS = SMG$$RECALC_PP_FIELDS (.DCB))
: 1206      1461  3          THEN
: 1207      1462  3              RETURN (.STATUS);
: 1208      1463  3
: 1209      1464  3          !+
: 1210      1465  3          ! Remap all pasteboard buffers to which we are pasted, from the
: 1211      1466  3          ! bottom outward.
: 1212      1467  3          !-
: 1213      1468  3          CURR_PP = .DCB [DCB_A_PP_NEXT];
: 1214      1469  3          WHILE .CURR_PP NEQ DCB [DCB_A_PP_NEXT]
: 1215      1470  3          DO

```



```

: 1216      1471  4      BEGIN      ! Remap all pasteboards
: 1217      1472  4      LOCAL
: 1218      1473  4      PBCB : REF $PBCB_DECL; ! Addr of a pasteboard control
: 1219      1474  4      ! block
: 1220      1475  4      PBCB = .CURR_PP [PP A PBCB ADDR];
: 1221      1476  5      IF NOT (STATUS = SMG$CHECK_FOR_OUTPUT_PBCB ( .PBCB))
: 1222      1477  4      THEN
: 1223      1478  4      RETURN ( .STATUS); ! Quit if any one of them fails
: 1224      1479  4
: 1225      1480  4      CURR_PP = .CURR_PP [PP A NEXT DCB]; ! To next pasting packet
: 1226      1481  3      END; ! Remap all pasteboards
: 1227      1482  3
: 1228      1483  3      RETURN ( SS$_NORMAL);
: 1229      1484  3      END ! Can do it now
: 1230      1485  3
: 1231      1486  3      ELSE
: 1232      1487  2
: 1233      1488  3      BEGIN ! Must delay until end_display_batch
: 1234      1489  3      DCB [DCB_V_PP MISMATCH] = 1; ! Mark it for later update
: 1235      1490  2      END; ! Must delay until end_display_batch
: 1236      1491  2
: 1237      1492  2      RETURN ( SS$_NORMAL);
: 1238      1493  1      END; ! Routine SMG$CHANGE_VIRTUAL_DISPLAY

```

			OFFC 00000	.ENTRY	SMG\$CHANGE_VIRTUAL_DISPLAY, Save R2,R3,R4,-	1016
					R5,R6,R7,R8,R9,R10,R11	
50	5E	30	C2 00002	SUBL2	#48, SP	
	6C	01	83 00005	SUBB3	#1, (AP), DIFF	1134
	05	50	91 00009	CMPB	DIFF, #5	
		08	1B 0000C	BLEQU	1\$	
	50	00000000G	8F D0 0000E	MOVL	#SMG\$_WRONUMARG, R0	
			04 00015	RET		
	50	04	BC D0 00016 1\$:	MOVL	@DISPLAY_ID, R0	1136
04	BC	38	A0 D1 0001A	CMPL	56(R0), @DISPLAY_ID	
		06	12 0001F	BNEQ	2\$	
	11	44	A0 91 00021	CMPB	68(R0), #17	
		08	13 00025	BEQL	3\$	
	50	00000000G	8F D0 00027 2\$:	MOVL	#SMG\$_INVDIS_ID, R0	
			04 0002E	RET		
	56	04	BC D0 0002F 3\$:	MOVL	@DISPLAY_ID, DCB	
	02		6C 91 00033	CMPB	(AP), #2	1142
		08	1F 00036	BLSSU	4\$	
		08	AC D5 00038	TSTL	8(AP)	
		06	13 0003B	BEQL	4\$	
	58	08	BC D0 0003D	MOVL	@NUM_ROWS, NEW_ROWS	1144
		04	11 00041	BRB	5\$	
	58	02	A6 3C 00043 4\$:	MOVZWL	2(DCB), NEW_ROWS	1146
	03		6C 91 00047 5\$:	CMPB	(AP), #3	1148
		0C	1F 0004A	BLSSU	6\$	
		0C	AC D5 0004C	TSTL	12(AP)	
		07	13 0004F	BEQL	6\$	
18	AE	0C	BC D0 00051	MOVL	@NUM_COLS, NEW_COLS	1150
		05	11 00056	BRB	7\$	



			18	AE	06	A6	3C	00058	6\$:	MOVZWL	6(DCB), NEW COLS	1152
	20	AE		58	18	AE	C5	0005D	7\$:	MULL3	NEW COLS, NEW_ROWS, NEW_SIZE	1154
				04		6C	91	00063		CMPB	(AP), #4	1160
						0A	1F	00066		BLSSU	8\$	
					10	AC	D5	00068		TSTL	16(AP)	
						05	13	0006B		BEQL	8\$	
		2F	A6		10	BC	90	0006D		MOVB	@DISPLAY_ATTRIBUTES, 47(DCB)	1162
			05			6C	91	00072	8\$:	CMPB	(AP), #5	1164
						0A	1F	00075		BLSSU	9\$	
					14	AC	D5	00077		TSTL	20(AP)	
						05	13	0007A		BEQL	9\$	
		2E	A6		14	BC	90	0007C		MOVB	@VIDEO_ATTRIBUTES, 46(DCB)	1166
			06			6C	91	00081	9\$:	CMPB	(AP), #6	1168
						0A	1F	00084		BLSSU	10\$	
					18	AC	D5	00086		TSTL	24(AP)	
						05	13	00089		BEQL	10\$	
		30	A6		18	BC	90	0008B		MOVB	@CHAR_SET, 48(DCB)	1170
			57		20	AE	D0	00090	10\$:	MOVL	NEW_SIZE, R7	1177
			57		3C	A6	D1	00094		CMPL	60(DCB), R7	
						14	12	00098		BNEQ	11\$	
58	02	A6		10		00	ED	0009A		CMPZV	#0, #16, 2(DCB), NEW_ROWS	1178
						0C	12	000A0		BNEQ	11\$	
18	AE	06	A6			00	ED	000A2		CMPZV	#0, #16, 6(DCB), NEW_COLS	1179
						03	12	000A9		BNEQ	11\$	
						01F7	31	000AB		BRW	29\$	
		18	AE		28	AE	9F	000AE	11\$:	PUSHAB	NEW_TEXT_BUF	1208
				57		01	78	000B1		ASHL	#1, R7, 24(SP)	
					18	AE	9F	000B6		PUSHAB	24(SP)	
			00000000G	00		02	FB	000B9		CALLS	#2, LIB\$GET_VM	
				6E		50	D0	000C0		MOVL	R0, STATUS	
				03		6E	E8	000C3		BLBS	STATUS, 12\$	
						016B	31	000C6		BRW	24\$	
		0C	AE		28	BE47	9E	000C9	12\$:	MOVAB	@NEW_TEXT_BUF[R7], NEW_ATTR_BUF	1213
57		20	6E			00	2C	000CF		MOVCS	#0, (SP), #32, R7, @NEW_TEXT_BUF	1222
					28	BE		000D4				
57	2E	A6		6E		00	2C	000D6		MOVCS	#0, (SP), 46(DCB), R7, @NEW_ATTR_BUF	1223
					0C	BE		000DC				
				5B	10	A6	D0	000DE		MOVL	16(DCB), TEXT_PTR	1225
				5A	14	A6	D0	000E2		MOVL	20(DCB), ATTR_PTR	1226
		04	AE		18	A6	D0	000E6		MOVL	24(DCB), CHAR_PTR	1227
			52		02	A6	3C	000EB		MOVZWL	2(DCB), R2	1229
			58			52	D1	000EF		CMPL	R2, NEW_ROWS	
						03	15	000F2		BLEQ	13\$	
				52		58	D0	000F4		MOVL	NEW_ROWS, R2	
		14	AE			52	D0	000F7	13\$:	MOVL	R2, ROWS_TO_MOVE	1230
			52		06	A6	3C	000FB		MOVZWL	6(DCB), R2	
		18	AE			52	D1	000FF		CMPL	R2, NEW_COLS	
						04	15	00103		BLEQ	14\$	
			52		18	AE	D0	00105		MOVL	NEW_COLS, R2	
		10	AE			52	D0	00109	14\$:	MOVL	R2, COLS_TO_MOVE	1247
						57	D4	0010D		CLRL	I	
						26	11	0010F		BRB	16\$	
			50		FF	A7	9E	00111	15\$:	MOVAB	-1(R7), R0	1239
			51		06	A6	3C	00115		MOVZWL	6(DCB), R1	
08	AE		50			51	C5	00119		MULL3	R1, R0, SOURCE_INDEX	
	59		50		0C	BC	C5	0011E		MULL3	@NUM_COLS, R0, DEST_INDEX	1240
28	BE49		08	BE4B	10	AE	28	00123		MOVCS	COLS_TO_MOVE, @SOURCE_INDEX[TEXT_PTR], -	1244



PC	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------



52	00	5A	24	AE	D0	00208	23\$:	MOVL	NEW_LINE_CHAR, R10	1352
		6E		00	2C	0020C		MOVC5	#0, (SP); #0, R2, (R10)	
	01	AA	01	A9	14	6A	00211			
			18	AE	4C	A6	00212	MOVC3	ROWS_TO_MOVE, 1(LINE_CHAR_PTR), 1(R10)	1360
					02	A6	00219	PUSHAB	76(DCB)	1367
					18	A6	0021C	MOVZWL	2(DCB), 24(SP)	1366
					18	AE	00221	INCL	24(SP)	
					18	AE	00224	PUSHAB	24(SP)	
	00000000G	00			02	FB	00227	CALLS	#2, LIB\$FREE_VM	1367
		6E			50	D0	0022E	MOVL	R0, STATUS	
		04			6E	E8	00231	BLBS	STATUS, 25\$	
		50			6E	D0	00234	24\$:	MOVL	STATUS, R0
						04	00237	RET		
	4C	A6		5A	D0	00238	25\$:	MOVL	R10, 76(DCB)	1374
	02	A6		58	B0	0023C	26\$:	MOVW	NEW_ROWS, 2(DCB)	1380
	06	A6	18	AE	B0	00240		MOVW	NEW_COLS, 6(DCB)	1381
	3C	A6	20	AE	D0	00245		MOVL	NEW_SIZE, 60(DCB)	1382
	28	A6	00010001	8F	D0	0024A		MOVL	#65537, 40(DCB)	1387
	34	A6		03	8A	00252		BICB2	#3, 52(DCB)	1395
	48	A6		01	B0	00256		MOVW	#1, 72(DCB)	1401
	4A	A6		58	B0	0025A		MOVW	NEW_ROWS, 74(DCB)	1402
		43		A6	E9	0025E		BLBC	47(DCB), 29\$	1407
		52		A6	9E	00262		MOVAB	8(R6), DESC	1414
			2F	A2	D5	00266		TSTL	4(DESC)	1415
			04	3A	13	00269		BEQL	29\$	
	2C	AE	2C	A6	3C	0026B		MOVZWL	44(DCB), TEMP	1420
	18	AE	33	A6	9A	00270		MOVZBL	51(DCB), 24(SP)	1439
			18	AE	9F	00275		PUSHAB	24(SP)	
04	34	A6		02	E1	00278		BBC	#2, 52(DCB), 27\$	1437
				7E	D4	0027D		CLRL	-(SP)	
				06	11	0027F		BRB	28\$	
		50	30	AE	9E	00281	27\$:	MOVAB	TEMP, R0	
				50	DD	00285		PUSHL	R0	
	1C	AE	31	A6	9A	00287	28\$:	MOVZBL	49(DCB), 28(SP)	1430
			1C	AE	9F	0028C		PUSHAB	28(SP)	
				52	DD	0028F		PUSHL	DESC	1429
			04	AC	DD	00291		PUSHL	DISPLAY_ID	1428
	0000V	CF		05	FB	00294		CALLS	#5, SMG\$LABEL_BORDER	
		09		50	E8	00299		BLBS	R0, 29\$	
	00000000G	00		52	DD	0029C		PUSHL	DESC	1442
			1C	01	FB	0029E		CALLS	#1, LIB\$SFREE1_DD	
				A6	D5	002A5	29\$:	TSTL	28(DCB)	1454
				2C	12	002A8		BNEQ	31\$	
				56	DD	002AA		PUSHL	DCB	1460
	0000V	CF		01	FB	002AC		CALLS	#1, SMG\$\$RECALC_PP_FIELDS	
		29		50	E9	002B1		BLBC	STATUS, 33\$	
		52	20	A6	D0	002B4		MOVL	32(DCB), CURR_PP	1468
		51	20	A6	9E	002B8	30\$:	MOVAB	32(DCB), R1	1469
		51		52	D1	002BC		CMPL	CURR_PP, R1	
				19	13	002BF		BEQL	32\$	
		51	14	A2	D0	002C1		MOVL	20(CURR_PP), PBCB	1475
				51	DD	002C5		PUSHL	PBCB	1476
	00000000G	00		01	FB	002C7		CALLS	#1, SMG\$\$CHECK_FOR_OUTPUT_PBCB	
		0C		50	E9	002CE		BLBC	STATUS, 33\$	
		52		62	D0	002D1		MOVL	(CURR_PP), CURR_PP	1480
				E2	11	002D4		BRB	30\$	1469
	34	A6		08	88	002D6	31\$:	BISB2	#8, 52(DCB)	1489



SMG\$DISPLAY\_LIN 1-096 SMG\$DISPLAY\_LINKS - Virtual Display Linkages 16-Sep-1984 00:29:22 VAX-11 Bliss-32 V4.0-742  
SMG\$CHANGE\_VIRTUAL\_DISPLAY - Change Virtual Dis 14-Sep-1984 13:09:43 [SMGRTL.SRC]SMGDISLIN.B32;1

Page 36  
(7)

50 01 D0 002DA 32\$: MOVL #1, R0  
04 002DD 33\$: RET

; 1492  
; 1493

; Routine Size: 734 bytes, Routine Base: \_SMG\$CODE + 0351

; 1239 1494 1 !<BLF/PAGE>



```

: 1241 1495 1 %SBTTL 'SMG$CHECK FOR OCCLUSION - Check to see if display is occluded'
: 1242 1496 1 GLOBAL ROUTINE SMG$CHECK_FOR_OCCLUSION (
: 1243 1497 1     DISPLAY ID,
: 1244 1498 1     PASTEBOARD ID,
: 1245 1499 1     OCCLUSION_STATE
: 1246 1500 1 ) =
: 1247 1501 1
: 1248 1502 1 ++
: 1249 1503 1 FUNCTIONAL DESCRIPTION:
: 1250 1504 1     This procedure determines if the given virtual display, as
: 1251 1505 1     pasted to the given pasteboard, is occluded by another virtual
: 1252 1506 1     display. The OCCLUSION state is set to:
: 1253 1507 1
: 1254 1508 1         1 : if virtual display is occluded
: 1255 1509 1         0 : if virtual display is not occluded.
: 1256 1510 1     not meaningfull : if status is not SS$NORMAL.
: 1257 1511 1
: 1258 1512 1     The returned status reflects whether the question could be
: 1259 1513 1     answered at all.
: 1260 1514 1
: 1261 1515 1 CALLING SEQUENCE:
: 1262 1516 1
: 1263 1517 1     ret_status.wlc.v = SMG$CHECK_FOR_OCCLUSION (
: 1264 1518 1         DISPLAY ID.rl.r,
: 1265 1519 1         PASTEBOARD ID.rl.r,
: 1266 1520 1         OCCLUSION_STATE.wl.r)
: 1267 1521 1
: 1268 1522 1 FORMAL PARAMETERS:
: 1269 1523 1     DISPLAY_ID.rl.r     Address of a display id.
: 1270 1524 1     PASTEBOARD_ID.rl.r  Address of a pasteboard id.
: 1271 1525 1     OCCLUSION_STATE.wl.r Set to      1 if occluded
: 1272 1526 1                                     0 if not occluded
: 1273 1527 1
: 1274 1528 1
: 1275 1529 1 IMPLICIT INPUTS:
: 1276 1530 1
: 1277 1531 1     NONE
: 1278 1532 1
: 1279 1533 1 IMPLICIT OUTPUTS:
: 1280 1534 1
: 1281 1535 1     NONE
: 1282 1536 1
: 1283 1537 1 COMPLETION STATUS:
: 1284 1538 1
: 1285 1539 1     SS$NORMAL      Normal success. OCCLUSION_STATE calculated.
: 1286 1540 1
: 1287 1541 1     SMG$_NOTPASTED Given virtual display is not pasted to given
: 1288 1542 1     pasteboard.
: 1289 1543 1
: 1290 1544 1     SMG$_INVPAS_ID Invalid pasteboard id.
: 1291 1545 1
: 1292 1546 1     SMG$_INVDIS_ID Invalid display id.
: 1293 1547 1
: 1294 1548 1
: 1295 1549 1 SIDE EFFECTS:
: 1296 1550 1
: 1297 1551 1

```



```

: 1298      1552  1  !      NONE
: 1299      1553  1  !--
: 1300      1554  1
: 1301      1555  2      BEGIN
: 1302      1556  2      LOCAL
: 1303      1557  2      STATUS,          ! Status of subroutine calls
: 1304      1558  2
: 1305      1559  2      PP : REF $PP_DECL,      ! Address of relevant pasting packet
: 1306      1560  2
: 1307      1561  2      PBCB: REF $PBCB_DECL,    ! Address of Pasteboard Control Block
: 1308      1562  2
: 1309      1563  2      DCB : REF $DCB_DECL;     ! Address of Display Control Block
: 1310      1564  2
: 1311      1565  2  !+
: 1312      1566  2  ! Validate number of arguments.
: 1313      1567  2  !-
: 1314      1568  2      $SMG$VALIDATE_ARGCOUNT (3, 3);
: 1315      1569  2
: 1316      1570  2  !+
: 1317      1571  2  ! Get DCB and PBCB addresses that go with these display ids and
: 1318      1572  2  ! pasteboard ids.
: 1319      1573  2  !-
: 1320      1574  2      $SMG$GET_DCB (.DISPLAY_ID, DCB);
: 1321      1575  2      $SMG$GET_PBCB (.PASTEBOARD_ID, PBCB);
: 1322      1576  2
: 1323      1577  2  !+
: 1324      1578  2  ! Try to find the pasting packet that binds these two. Return
: 1325      1579  2  ! SMG$_NOTPASTED it can't be located.
: 1326      1580  2  !-
: 1327      1581  3      IF NOT (STATUS = SMG$$LOCATE_PP ( .DCB, .PBCB, PP))
: 1328      1582  2      THEN
: 1329      1583  2      RETURN (.STATUS);
: 1330      1584  2
: 1331      1585  2  !+
: 1332      1586  2  ! Check to see if occluded and return appropriate OCCLUSION_STATE.
: 1333      1587  2  !-
: 1334      1588  3      .OCCLUSION_STATE = ( IF .PP [PP_V_OCCLUDED] THEN 1      ! Occluded
: 1335      1589  2      ELSE 0);      ! Not occluded
: 1336      1590  2
: 1337      1591  2      RETURN SS$_NORMAL;
: 1338      1592  1      END;          ! End of routine SMG$CHECK_FOR_OCCLUSION

```

				0004 00000	.ENTRY	SMG\$CHECK FOR OCCLUSION, Save R2	: 1496
52	00000000'	EF	9E	00002	MOVAB	PBD_L COUNT, R2	:
5E		04	C2	00009	SUBL2	#4, SP	:
03		6C	91	0000C	CMPB	(AP), #3	: 1568
		08	13	0000F	BEQL	1\$	:
50	00000000G	8F	D0	00011	MOVL	#SMG\$_WRONUMARG, R0	:
				04 00018	RET		:
50		04	BC	D0 00019	MOVL	@DISPLAY ID, R0	: 1574
04	BC	38	A0	D1 0001D	CMPL	56(R0), @DISPLAY_ID	:
		06	12	00022	BNEQ	2\$	:
11		44	A0	91 00024	CMPB	68(R0), #17	:



			08	13	00028	BEQL	3\$		
	50	00000000G	8F	D0	0002A	2\$:	MOVL	#SMG\$_INVDIS_ID, R0	
				04	00031		RET		
	51	04	BC	D0	00032	3\$:	MOVL	@DISPLAY_ID, DCB	
	50	08	BC	D0	00036		MOVL	@PASTEBOARD_ID, R0	1575
				0A	19	0003A	BLSS	4\$	
	62		50	D1	0003C		CMPL	R0, PBD_L_COUNT	
				05	14	0003F	BGTR	4\$	
08	44	A2	50	E0	00041		BBS	R0, PBD_V_PB_AVAIL, 5\$	
			50	00000000G	8F	4\$:	MOVL	#SMG\$_INVPAS_ID, R0	
				04	0004D		RET		
			50	04	A240	5\$:	MOVL	PBD A PBCB[R0], PBCB	
				4001	8F		PUSHR	#^M<R0, SP>	1581
				51	DD	00057	PUSHL	DCB	
0000V		CF		03	FB	00059	CALLS	#3, SMG\$\$LOCATE_PP	
		15		50	E9	0005E	BLBC	STATUS, 8\$	
		50		6E	D0	00061	MOVL	PP, R0	1588
		05	2A	A0	E9	00064	BLBC	42(R0), 6\$	
		50		01	D0	00068	MOVL	#1, R0	
				02	11	0006B	BRB	7\$	
				50	D4	0006D	6\$:	CLRL	R0
	0C	BC		50	D0	0006F	7\$:	MOVL	R0, @OCCLUSION_STATE
		50		01	D0	00073	MOVL	#1, R0	1591
				04	00076	8\$:	RET		1592

; Routine Size: 119 bytes, Routine Base: \_SMG\$CODE + 062F

; 1339 1593 1 !<BLF/PAGE>



```

: 1341 1594 1 %SBTTL 'SMG$CREATE_PASTEBOARD - Create Pasteboard'
: 1342 1595 1 GLOBAL ROUTINE SMG$CREATE_PASTEBOARD (
: 1343 1596 1     NEW_PBID,
: 1344 1597 1     OUT_DEVICE,
: 1345 1598 1     PB_ROWS,
: 1346 1599 1     PB_COLS,
: 1347 1600 1     PRESERVE_SCREEN_FLAG
: 1348 1601 1 ) =
: 1349 1602 1
: 1350 1603 1 ++
: 1351 1604 1 FUNCTIONAL DESCRIPTION:
: 1352 1605 1     This routine creates a new pasteboard -- returning its assigned
: 1353 1606 1     pasteboard_id. OUT_DEVICE is the device upon which this
: 1354 1607 1     pasteboard is to be written. If not supplied, output will flow
: 1355 1608 1     to SYS$OUTPUT.
: 1356 1609 1
: 1357 1610 1     If PB_ROWS and/or PB_COLS are provided, they are filled in with
: 1358 1611 1     the number of rows and number of columns on the physical device.
: 1359 1612 1
: 1360 1613 1     If called upon to create a 2nd pasteboard on a device that
: 1361 1614 1     already has a pasteboard associated with it, we simply return
: 1362 1615 1     the id of the already-existing pasteboard and the qualified
: 1363 1616 1     success SMG$_PASALREXI.
: 1364 1617 1
: 1365 1618 1 CALLING SEQUENCE:
: 1366 1619 1
: 1367 1620 1     ret_status.wlc.v = SMG$CREATE_PASTEBOARD (
: 1368 1621 1         NEW_PBID.wl.r
: 1369 1622 1         [,OUT_DEVICE.rt.dx]
: 1370 1623 1         [,PB_ROWS.wl.r]
: 1371 1624 1         [,PB_COLS.wl.r]
: 1372 1625 1         [,PRESERVE_SCREEN_FLAG.rl.r])
: 1373 1626 1
: 1374 1627 1 FORMAL PARAMETERS:
: 1375 1628 1
: 1376 1629 1     NEW_PBID.wl.r     Pasteboard id of newly-created pasteboard.
: 1377 1630 1
: 1378 1631 1     OUT_DEVICE.rt.dx  [Optional]. If supplied, this parameter
: 1379 1632 1                       is the file specification or logical
: 1380 1633 1                       name upon which the output associated
: 1381 1634 1                       with this pasteboard will be written.
: 1382 1635 1                       If omitted, output goes to SYS$OUTPUT.
: 1383 1636 1
: 1384 1637 1     PB_ROWS           [Optional]. If provided, it is filled in with
: 1385 1638 1                       the number of rows on the physical device.
: 1386 1639 1
: 1387 1640 1     PB_COLS           [Optional]. If provided, it is filled in with
: 1388 1641 1                       the number of columns on the physical device.
: 1389 1642 1
: 1390 1643 1     PRESERVE_SCREEN_FLAG [Optional]. If provided, and if has a
: 1391 1644 1                       value of 1, then the screen will not
: 1392 1645 1                       be initially cleared.
: 1393 1646 1
: 1394 1647 1 IMPLICIT INPUTS:
: 1395 1648 1
: 1396 1649 1     NONE
: 1397 1650 1

```



SMG\$DISPLAY\_LIN SMG\$DISPLAY LINKS - Virtual Display Linkages  
1-096 SMG\$CREATE\_PASTEBOARD - Create Pasteboard

L 14  
16-Sep-1984 00:29:22  
14-Sep-1984 13:09:43

VAX-11 Bliss-32 V4.0-742  
[SMGRTL.SRC]SMGDISLIN.B32;1

Page 41  
(9)

```
: 1398      1651  1  |
: 1399      1652  1  | IMPLICIT OUTPUTS:
: 1400      1653  1  |
: 1401      1654  1  |     NONE
: 1402      1655  1  |
: 1403      1656  1  | COMPLETION STATUS:
: 1404      1657  1  |
: 1405      1658  1  |     $$$ NORMAL      Normal successful completion
: 1406      1659  1  |     LIB$_INSVIRMEM  Insufficient virtual memory to allocate needed
: 1407      1660  1  |                      buffer.
: 1408      1661  1  |     SMG$_PASALREXI  Pasteboard already exists for this device
: 1409      1662  1  |     SMG$_WRONUMARG  Wrong number of arguments.
: 1410      1663  1  |
: 1411      1664  1  | SIDE EFFECTS:
: 1412      1665  1  |
: 1413      1666  1  |     NONE
: 1414      1667  1  | --
```



```

: 1416      1668 2      BEGIN
: 1417      1669 2
: 1418      1670 2      BUILTIN
: 1419      1671 2          NULLPARAMETER;
: 1420      1672 2
: 1421      1673 2      LOCAL
: 1422      1674 2          FS_LEN : WORD INITIAL (0),      ! Length of filespec name to use.
: 1423      1675 2
: 1424      1676 2          FS_ADDR,      ! Address of filespec name to use.
: 1425      1677 2
: 1426      1678 2          STATUS,      ! Status of subroutine calls
: 1427      1679 2
: 1428      1680 2          TERM_TYPE,      ! terminal type
: 1429      1681 2
: 1430      1682 2          CLEAR_FLAG,      ! TRUE means clear screen
: 1431      1683 2
: 1432      1684 2          PBID,      ! Id of pasteboard being created.
: 1433      1685 2
: 1434      1686 2          PBCB : REF $PBCB_DECL;      ! Address of pasteboard control block
: 1435      1687 2          ! being created.
: 1436      1688 2
: 1437      1689 2      EXTERNAL ROUTINE
: 1438      1690 2
: 1439      1691 2          SMG$$ERASE_PASTEBOARD,
: 1440      1692 2          SMG$$OUT_OF_BAND_HANDLER;
: 1441      1693 2
: 1442      1694 2      $SMG$VALIDATE_ARGCOUNT (1, 5);      ! Test for right no. of args
: 1443      1695 2
: 1444      1696 2      $SMG$GET_NEXT_PID ( PBID);      ! Allocate a new PBID
: 1445      1697 2
: 1446      1698 2      !+
: 1447      1699 2      ! Decide what output device is to receive the the output of this
: 1448      1700 2      ! pasteboard.
: 1449      1701 2      !-
: 1450      1702 2      FS_LEN = %CHARCOUNT ('SYSS$OUTPUT');      ! Assume default
: 1451      1703 2      FS_ADDR = UPLIT (BYTE ('SYSS$OUTPUT'));
: 1452      1704 2
: 1453      1705 2      IF NOT NULLPARAMETER (OUT_DEVICE)
: 1454      1706 2      THEN
: 1455      1707 3          BEGIN      ! User-supplied filespec
: 1456      1708 4              IF NOT (STATUS = LIB$ANALYZE_SDESC_R2 ( .OUT_DEVICE ;
: 1457      1709 4                  FS_LEN, FS_ADDR))
: 1458      1710 3                  THEN
: 1459      1711 3                      RETURN ( .STATUS);
: 1460      1712 2                  END;      ! User-supplied filespec
: 1461      1713 2
: 1462      1714 2      !+
: 1463      1715 2      ! Create a PBCB. Allocate buffers, etc.
: 1464      1716 2      ! Extract the necessary device attributes and store in PBCB.
: 1465      1717 2      !-
: 1466      1718 2      STATUS = SMG$$SETUP TERMINAL_TYPE (
: 1467      1719 2          .FS_ADDR,      ! filespec addr
: 1468      1720 2          .FS_LEN,      ! Len of filespec
: 1469      1721 2          TERM_TYPE,      ! Gets terminal type
: 1470      1722 2          PBCB;      ! Address to receive address of PBCB
: 1471      1723 2
: 1472      1724 2      IF NOT .STATUS

```



```

1473 1725 2 THEN
1474 1726 3 BEGIN
1475 1727 3 PBD V PB_AVAIL [.PBID] = 0; ! Release PBID number
1476 1728 4 RETURN (.STATUS)
1477 1729 2 END;
1478 1730 2
1479 1731 2 !+
1480 1732 2 Decide whether we want to handle this output device ourselves
1481 1733 2 or use RMS to handle it. We use RMS if the output device is
1482 1734 2 not a terminal. We also use RMS if the output device is a terminal,
1483 1735 2 but one we can't handle, such as a hardcopy terminal.
1484 1736 2 -
1485 1737 2
1486 1738 2 PBCB [PBCB_V_RMS] = (.PBCB[PBCB_B_CLASS] NEQ DC$ TERM ) OR
1487 1739 2 (.PBCB[PBCB_B_DEVTTYPE] EQL UNKNOWN ) OR
1488 1740 2 (.PBCB[PBCB_B_DEVTTYPE] EQL HARDCOPY );
1489 1741 2
1490 1742 2 !+
1491 1743 2 Loop through all the pasteboards we currently have trying to find
1492 1744 2 one whose associated resultant name string is the same as the one we
1493 1745 2 just created.
1494 1746 2 If we can find one, we have just created a 2nd pasteboard for the same
1495 1747 2 physical device and we want to get rid of the pasteboard we just
1496 1748 2 created and return to the caller the id of the pasteboard that already
1497 1749 2 exists for this device.
1498 1750 2 We do this only if the output device is a terminal.
1499 1751 2 If the output device is a file, we assume that the user wants
1500 1752 2 to create a new file for each pasteboard he creates.
1501 1753 2 -
1502 1754 2
1503 1755 2 IF NOT .PBCB [PBCB_V_RMS]
1504 1756 2 THEN
1505 1757 2 INCR I FROM 0 TO .PBD_L_COUNT -1
1506 1758 2 DO
1507 1759 2 BEGIN ! Loop thru pasteboards
1508 1760 2 LOCAL
1509 1761 2 SEARCH_PBCB : REF $PBCB_DECL; ! Addr of pasteboard
1510 1762 2 ! control blocks that
1511 1763 2 ! we are inspecting.
1512 1764 2
1513 1765 2 IF (SEARCH_PBCB = .PBD_A_PBCB [.I]) NEQ 0
1514 1766 2 THEN
1515 1767 2 BEGIN ! A valid pasteboard address
1516 1768 2 IF .SEARCH_PBCB [PBCB_W_DEVNAM_LEN] EQL
1517 1769 2 .PBCB [PBCB_W_DEVNAM_LEN]
1518 1770 2 THEN
1519 1771 2 BEGIN ! Lengths match
1520 1772 2 IF CH$EQL ( .SEARCH_PBCB [PBCB_W_DEVNAM_LEN], ! length
1521 1773 2 SEARCH_PBCB [PBCB_T_DEVNAM], ! addr
1522 1774 2 .PBCB [PBCB_W_DEVNAM_LEN], ! length
1523 1775 2 PBCB [PBCB_T_DEVNAM]) ! addr
1524 1776 2 THEN
1525 1777 2 BEGIN ! Match found
1526 1778 2 LOCAL
1527 1779 2 STATUS; ! Local status of subr. calls
1528 1780 2
1529 1781 2 !+

```



```

: 1530      1782  6      | This physical device already has a pasteboard
: 1531      1783  6      | associated with it.
: 1532      1784  6      | Get rid of the one we just created.
: 1533      1785  6      | First return PBID number we consumed -- we won't
: 1534      1786  6      | be using it.
: 1535      1787  6      |
: 1536      1788  6      | PBD_V_PB_AVAIL [.PBID] = 0;
: 1537      1789  6      |
: 1538      1790  6      | +
: 1539      1791  6      | Second deallocate the WCB that got allocated.
: 1540      1792  6      |
: 1541      1793  6      | IF .PBCB [PBCB_A_WCB] NEQ 0
: 1542      1794  6      | THEN
: 1543      1795  7      |     IF NOT (STATUS = SMG$$DEALLOCATE_WCB (
: 1544      1796  7      |         .PBCB [PBCB_A_WCB]) )
: 1545      1797  6      |     THEN
: 1546      1798  6      |         RETURN (.STATUS);
: 1547      1799  6      |
: 1548      1800  6      | +
: 1549      1801  6      | Next release output buffer.
: 1550      1802  6      |
: 1551      1803  6      | IF .PBCB [PBCB_A_OUTPUT_BUFFER] NEQ 0
: 1552      1804  6      | THEN
: 1553      1805  7      |     IF NOT ( STATUS = LIB$FREE_VM (
: 1554      1806  7      |         %REF (.PBCB [PBCB_W_OUTPUT_BUFSIZ]),
: 1555      1807  7      |         PBCB [PBCB_A_OUTPUT_BUFFER]) )
: 1556      1808  6      |     THEN
: 1557      1809  6      |         RETURN (.STATUS);
: 1558      1810  6      |
: 1559      1811  6      | +
: 1560      1812  6      | Finally release the PBCB itself.
: 1561      1813  6      |
: 1562      1814  7      | IF NOT (STATUS = LIB$FREE_VM ( %REF (PBCB_K_SIZE),
: 1563      1815  7      |         PBCB))
: 1564      1816  6      | THEN
: 1565      1817  6      |     RETURN (.STATUS);
: 1566      1818  6      |
: 1567      1819  6      | +
: 1568      1820  6      | Return as an id the id of the one that already
: 1569      1821  6      | exists.
: 1570      1822  6      |
: 1571      1823  6      | .NEW_PBID = .SEARCH_PBCB [PBCB_L_PBID];
: 1572      1824  6      |
: 1573      1825  6      | +
: 1574      1826  6      | If caller requested number of rows and columns on
: 1575      1827  6      | device, tell him.
: 1576      1828  6      |
: 1577      1829  6      | IF NOT NULLPARAMETER (PB_ROWS)
: 1578      1830  6      | THEN .PB_ROWS = .SEARCH_PBCB [PBCB_B_ROWS];
: 1579      1831  6      |
: 1580      1832  6      | IF NOT NULLPARAMETER (PB_COLS)
: 1581      1833  6      | THEN .PB_COLS = .SEARCH_PBCB [PBCB_W_WIDTH];
: 1582      1834  6      |
: 1583      1835  6      | RETURN ( SMG$ _PASALREXI );
: 1584      1836  5      | END;      ! Match found
: 1585      1837  4      | END;      ! Lengths match
: 1586      1838  3      | END;      ! A valid pasteboard address

```



```

1587 1839 2      END;      ! Loop thru pasteboards
1588 1840 2
1589 1841 2
1590 1842 2      +
1591 1843 2      ! If we fall out of loop, none of our current pasteboards are pasted to
1592 1844 2      ! the same device. Continue with the creation process.
1593 1845 2      ! Store pasteboard id in the PBCB itself.
1594 1846 2      -
1595 1847 2      PBCB [PBCB_L_PBID] = .PBID;
1596 1848 2
1597 1849 2      +
1598 1850 2      ! Store the original name (that the user specified) for this device
1599 1851 2      ! in the PBCB. This name may include a filename as well as a
1600 1852 2      ! device name.
1601 1853 2      ! First we allocate virtual memory for this buffer and
1602 1854 2      ! then we store the length and address in the PBCB for future reference.
1603 1855 2      -
1604 1856 2      STATUS=LIB$GET_VM(%REF(.FS_LEN),PBCB[PBCB_A_OUTNAM]);
1605 1857 2      IF NOT .STATUS THEN RETURN(.STATUS);
1606 1858 2      PBCB[PBCB_W_OUTNAM_LEN]=.FS_LEN;
1607 1859 2      CH$MOVE(.FS_LEN,.FS_ADDR,.PBCB[PBCB_A_OUTNAM]);
1608 1860 2
1609 1861 2      +
1610 1862 2      ! If device is a terminal, assign a channel to it.
1611 1863 2      ! If the device is not a terminal, allocate a FAB and RAB
1612 1864 2      ! and open the file for output using RMS.
1613 1865 2      -
1614 1866 2
1615 1867 2      IF .PBCB[PBCB_V_RMS]
1616 1868 2      THEN BEGIN ! use RMS to open output
1617 1869 2
1618 1870 2      +
1619 1871 2      ! Allocate a FAB and RAB to be used to talk to this file.
1620 1872 2      -
1621 1873 2
1622 1874 2      STATUS=LIB$GET_VM(%REF(FAB$C_BLN),PBCB[PBCB_A_FAB]);
1623 1875 2      IF NOT .STATUS THEN RETURN .STATUS;
1624 1876 2
1625 1877 2      STATUS=LIB$GET_VM(%REF(RAB$C_BLN),PBCB[PBCB_A_RAB]);
1626 1878 2      IF NOT .STATUS THEN RETURN .STATUS;
1627 1879 2
1628 1880 2      +
1629 1881 2      ! Allocate a record buffer.
1630 1882 2      ! This will be one byte larger than the width of
1631 1883 2      ! the pasteboard because sometimes we will prepend
1632 1884 2      ! a formfeed to the record.
1633 1885 2      -
1634 1886 2
1635 1887 2      STATUS=LIB$GET_VM(%REF(.PBCB[PBCB_W_WIDTH]+1),PBCB[PBCB_A_RBF]);
1636 1888 2      IF NOT .STATUS THEN RETURN .STATUS;
1637 1889 2
1638 1890 2      +
1639 1891 2      ! Initialize the FAB and RAB.
1640 1892 2      -
1641 1893 2
1642 1894 2      $FAB_INIT(      FAB      = .PBCB[PBCB_A_FAB],
1643 1895 2      P      DNM      = 'SMGOUTPUT.LIS',      ! default filename

```



```

: 1644 P 1896 3 CT) = .PBCB, ! why not? pass the PBCB as user context
: 1645 P 1897 FAC = PUT, ! write access only
: 1646 P 1898 FNA = .FS_ADDR,
: 1647 P 1899 FNS = .FS_LEN,
: 1648 P 1900 ORG = SEQ, ! sequential file
: 1649 P 1901 FOP = SQO, ! sequential operations only
: 1650 P 1902 RAT = CR, ! carriage control
: 1651 P 1903 RFM = VAR, ! variable length records
: 1652 1904 MRS = .PBCB[PBCB_W_WIDTH+1]; ! max record size
: 1653 1905
: 1654 P 1906 $RAB_INIT( RAB = .PBCB[PBCB_A_RAB],
: 1655 P 1907 CTX = .PBCB, ! pass the PBCB as user context
: 1656 P 1908 FAB = .PBCB[PBCB_A_FAB],
: 1657 P 1909 RBF = .PBCB[PBCB_A_RBF],
: 1658 1910 RAC = SEQ); ! sequential output
: 1659 1911
: 1660 1912 !+
: 1661 1913 ! Open the file for output.
: 1662 1914 !-
: 1663 1915
: 1664 1916 STATUS=$CREATE( FAB = .PBCB[PBCB_A_FAB]);
: 1665 1917 IF NOT .STATUS THEN RETURN .STATUS;
: 1666 1918
: 1667 1919 !+
: 1668 1920 ! Connect a record stream to the file.
: 1669 1921 !-
: 1670 1922
: 1671 1923 STATUS=$CONNECT( RAB = .PBCB[PBCB_A_RAB]);
: 1672 1924 IF NOT .STATUS THEN RETURN .STATUS;
: 1673 1925
: 1674 1926 ELSE END ! use RMS to open output
: 1675 1927 BEGIN ! assigning channel
: 1676 1928
: 1677 1929 LOCAL NAME_DESC : VECTOR[2], ! Fixed length descriptor
: 1678 1930 ASYNC_EFN : LONG, ! Longword to hold efn
: 1679 1931 TTIOB : VECTOR[4,WORD], ! IOB for SENSE MODE
: 1680 1932 CHARBUF : BLOCK[12,BYTE]; ! 12-byte characteristics buffer
: 1681 1933
: 1682 1934 !+
: 1683 1935 ! Create a fixed length descriptor for our device name string
: 1684 1936 ! for use by $ASSIGN.
: 1685 1937 !-
: 1686 1938
: 1687 1939 NAME_DESC[0]=.PBCB[PBCB_W_DEVNAM_LEN];
: 1688 1940 NAME_DESC[1]=.PBCB[PBCB_T_DEVNAM];
: 1689 1941
: 1690 1942 !+
: 1691 1943 ! Assign the channel.
: 1692 1944 ! Put the resulting channel number in PBCB[PBCB_W_CHAN].
: 1693 1945 !-
: 1694 1946
: 1695 P 1947 STATUS=$ASSIGN( DEVNAM = NAME_DESC,
: 1696 1948 CHAN = PBCB[PBCB_W_CHAN]);
: 1697 1949 IF NOT .STATUS THEN RETURN .STATUS;
: 1698 1950
: 1699 1951 !+
: 1700 1952 ! Assign an asynchronous event flag.

```



```

: 1701      1953      3      !-
: 1702      1954      3
: 1703      1955      3      STATUS=LIB$GET_EF(ASYNC_EFN);
: 1704      1956      3      IF NOT .STATUS THEN RETURN .STATUS;
: 1705      1957      3
: 1706      1958      3      !+
: 1707      1959      3      ! Store the value into a byte in the PBCB.
: 1708      1960      3      !-
: 1709      1961      3
: 1710      1962      3      PBCB [PBCB_B_ASYNC_EFN] = .ASYNC_EFN;
: 1711      1963      3
: 1712      1964      3      !+
: 1713      1965      3      ! Do a SENSE MODE QIO to get additional characteristics
: 1714      1966      3      ! of interest.
: 1715      1967      3      ! Ignore everything returned in the characteristics buffer.
: 1716      1968      3      ! (We already got that stuff.)
: 1717      1969      3      ! The I/O status block has neat things of interest.
: 1718      1970      3      !-
: 1719      1971      3
: 1720      1972      3      STATUS=$QIOW(  CHAN      = .PBCB[PBCB_W_CHAN],
: 1721      1973      3      FUNC      = IO$_SENSEMODE,
: 1722      1974      3      IOSB      = TTIOSB,
: 1723      1975      3      P1        = CHARBUF,
: 1724      1976      3      P2        = 12);
: 1725      1977      3      IF NOT .STATUS THEN RETURN .STATUS;
: 1726      1978      3      IF NOT .TTIOSB[0] THEN RETURN .TTIOSB[0];
: 1727      1979      3
: 1728      1980      3      PBCB [PBCB_W_SPEED] = .TTIOSB[1];
: 1729      1981      3      PBCB [PBCB_W_FILL]   = .TTIOSB[2];
: 1730      1982      3      PBCB [PBCB_B_PARITY] = .TTIOSB[3];
: 1731      1983      3
: 1732      1984      3      END;      ! assigning channel
: 1733      1985      3
: 1734      1986      3      !+
: 1735      1987      3      ! Set up our exit block which is contained within the PBCB.
: 1736      1988      3      ! This exit block is used to establish an exit handler for
: 1737      1989      3      ! this terminal. When the exit handler is called,
: 1738      1990      3      ! it will flush the output buffers.
: 1739      1991      3      ! This guarantees that the user will see all his output even if
: 1740      1992      3      ! his program exits and he doesn't manually flush the buffers.
: 1741      1993      3      !-
: 1742      1994      3
: 1743      1995      3      PBCB [PBCB_A_EXIT_ADDR] = SMG$$PBCB_EXIT_HANDLER;
: 1744      1996      3      PBCB [PBCB_B_EXIT_ARGCNT] = 2;
: 1745      1997      3      PBCB [PBCB_A_EXIT_RSN] = PBCB [PBCB_L_EXIT_REASON];
: 1746      1998      3
: 1747      1999      3
: 1748      2000      3
: 1749      2001      3
: 1750      2002      3
: 1751      2003      3
: 1752      2004      3
: 1753      2005      3      PBCB [PBCB_A_EXIT_PBCB] = .PBCB;
: 1754      2006      3
: 1755      2007      3
: 1756      2008      3
: 1757      2009      3

```

```

      Address of our exit handler
      Our exit handler gets called with
      two arguments.
      The first argument is the address
      of the longword to receive the
      exit reason. This longword appears
      elsewhere in the PBCB (not in
      the exit block).
      The second argument is the address
      of this PBCB. This is needed
      because there are many PBCBs and
      one exit routine serves them all.
      There is a separate exit block for

```



```

: 1758      2010      2
: 1759      2011      2
: 1760      2012      2
: 1761      2013      2
: 1762      2014      2
: 1763      2015      2
: 1764      2016      2
: 1765      2017      2
: 1766      2018      2
: 1767      2019      2
: 1768      2020      2
: 1769      2021      2
: 1770      2022      2
: 1771      2023      2
: 1772      2024      2
: 1773      2025      2
: 1774      2026      2
: 1775      2027      2
: 1776      2028      2
: 1777      2029      2
: 1778      2030      2
: 1779      2031      2
: 1780      2032      2
: 1781      2033      2
: 1782      2034      2
: 1783      2035      2
: 1784      2036      2
: 1785      2037      2
: 1786      2038      2
: 1787      2039      2
: 1788      2040      2
: 1789      2041      2
: 1790      2042      2
: 1791      2043      2
: 1792      2044      2
: 1793      2045      2
: 1794      2046      2
: 1795      2047      2
: 1796      2048      2
: 1797      2049      2
: 1798      2050      2
: 1799      2051      2
: 1800      2052      2
: 1801      2053      2
: 1802      2054      2
: 1803      2055      2
: 1804      2056      2
: 1805      2057      2
: 1806      2058      2
: 1807      2059      2
: 1808      2060      2
: 1809      2061      2
: 1810      2062      2
: 1811      2063      2
: 1812      2064      2
: 1813      2065      2
: 1814      2066      2

! each pasteboard.

+ Establish the exit handler, using the exit block just created.
-

STATUS=$DCLEXH(DESBK=PBCB [PBCB_R_EXIT_BLOCK]);
IF NOT .STATUS THEN RETURN .STATUS;

+
Now we do an incredible strange thing.
We build a 10-byte routine in the PBCB to service out-of-band ASTs.
The routine has the form:

      0000      entry mask
      FA      CALLG
      6C      (AP)
      9F      absolute addressing
address longword address of SMG$$OUT_OF_BAND_HANDLER
      04      RET

Symbolically, the routine looks as follows:

ROUTINE BAND_HANDLER =
BEGIN
EXTERNAL ROUTINE SMG$$OUT_OF_BAND_HANDLER : ADDRESSING_MODE(ABSOLUTE);
BUILTIN AP,CALLG;
RETURN CALLG(.AP,SMG$$OUT_OF_BAND_HANDLER);
END;

However, we don't actually create this routine in BLISS and then
move it into our structure, because we can't be guaranteed that
BLISS will continue to generate the same code in future releases.
Thus we create the entire routine ourselves.
This code would have to change if we ever tried to run this
on a machine with a new architecture.

-

PBCB[PBCB_W_ENTRY_MASK]      = %X'0000';
PBCB[PBCB_B_CALLG]           = %X 'FA';
PBCB[PBCB_B_REG_AP]          = %X '6C';
PBCB[PBCB_B_ABS]             = %X '9F';
PBCB[PBCB_A_BAND_HANDLER]    = SMG$$OUT_OF_BAND_HANDLER;
PBCB[PBCB_B_RET]             = %X '04';

+
Since all went well, we can now adjust the count of how many PBCB's
we have and plug its address into the pasteboard directory.
-

PBD_L_COUNT = .PBD_L_COUNT + 1;

PBD_A_PBCB [.PBID] = .PBCB;

+
Initially clear the screen (unless we are asked to preserve it).
-

```



```

: 1815      2067 2  CLEAR FLAG=1;
: 1816      2068 2  IF NOT NULLPARAMETER(PRESERVE_SCREEN_FLAG)
: 1817      2069 2  THEN CLEAR_FLAG=NOT ..PRESERVE_SCREEN_FLAG;
: 1818      2070 2
: 1819      2071 2  IF .CLEAR_FLAG
: 1820      2072 2  THEN
: 1821      2073 2  BEGIN
: 1822      2074 2  STATUS=SMG$$ERASE_PASTEBOARD(.PBCB);
: 1823      2075 2  IF NOT .STATUS THEN RETURN .STATUS;
: 1824      2076 2  END
: 1825      2077 2  ELSE
: 1826      2078 2  BEGIN ! Just pretend we cleared the screen.
: 1827      2079 2  LOCAL WCB : REF $WCB_DECL;
: 1828      2080 2  WCB=.PBCB[PBCB_A_WCB];
: 1829      2081 2
: 1830      2082 2  CH$FILL(%' ',..WCB[WCB_L_BUFSIZE],..WCB[WCB_A_TEXT_BUF]);
: 1831      2083 2  CH$FILL(%' ',..WCB[WCB_L_BUFSIZE],..WCB[WCB_A_SCR_TEXT_BUF]);
: 1832      2084 2  CH$FILL(0, ..WCB[WCB_L_BUFSIZE],..WCB[WCB_A_ATTR_BUF]);
: 1833      2085 2  CH$FILL(0, ..WCB[WCB_L_BUFSIZE],..WCB[WCB_A_SCR_ATTR_BUF]);
: 1834      2086 2
: 1835      2087 2  IF .WCB[WCB_A_CHAR_SET_BUF] NEQ 0
: 1836      2088 2  THEN
: 1837      2089 2  BEGIN
: 1838      2090 2  CH$FILL(0,..WCB[WCB_L_BUFSIZE],..WCB[WCB_A_CHAR_SET_BUF]);
: 1839      2091 2  END;
: 1840      2092 2
: 1841      2093 2  IF .WCB[WCB_A_SCR_CHAR_SET_BUF] NEQ 0
: 1842      2094 2  THEN
: 1843      2095 2  BEGIN
: 1844      2096 2  CH$FILL(0,..WCB[WCB_L_BUFSIZE],..WCB[WCB_A_SCR_CHAR_SET_BUF]);
: 1845      2097 2  END;
: 1846      2098 2
: 1847      2099 2  !+
: 1848      2100 2  ! The physical cursor moves to (1,1).
: 1849      2101 2  !-
: 1850      2102 2
: 1851      2103 2  ! WCB[WCB_W_CURR_CUR_ROW]=1;
: 1852      2104 2  ! WCB[WCB_W_OLD_CUR_ROW]=1;
: 1853      2105 2  ! WCB[WCB_W_CURR_CUR_COL]=1;
: 1854      2106 2  ! WCB[WCB_W_OLD_CUR_COL]=1;
: 1855      2107 2
: 1856      2108 2  !+
: 1857      2109 2  ! The line characteristics get set back to 0.
: 1858      2110 2  !-
: 1859      2111 2
: 1860      2112 2  CH$FILL(0,..WCB[WCB_W_NO_ROWS]+1,..WCB[WCB_A_LINE_CHAR]);
: 1861      2113 2  CH$FILL(0,..WCB[WCB_W_NO_ROWS]+1,..WCB[WCB_A_SCR_LINE_CHAR]);
: 1862      2114 2
: 1863      2115 2  END; ! Just pretend we cleared the screen
: 1864      2116 2
: 1865      2117 2  !+
: 1866      2118 2  ! If caller is interested in number of rows and columns on device, tell
: 1867      2119 2  ! him.
: 1868      2120 2  !-
: 1869      2121 2  IF NOT NULLPARAMETER(PB_ROWS) THEN .PB_ROWS = .PBCB [PBCB_B_ROWS];
: 1870      2122 2  IF NOT NULLPARAMETER(PB_COLS) THEN .PB_COLS = .PBCB [PBCB_W_WIDTH];
: 1871      2123 2

```



													006A6	.BLKB	2						:
53	49	4C	54 2E	55 54	50 50	54 54	55 54	4F 55	24 4F	53 47	59 4D	53 53	006A8 P.AAA: 006B2 P.AAB:	.ASCII .ASCII	\SYSS\$OUTPUT\ \SMGOUTPUT.LIS\					:	
														.EXTRN .EXTRN .EXTRN .EXTRN	SMG\$\$OUT OF BAND HANDLER SYSS\$CREATE,-SYSS\$CONNECT SYSS\$ASSIGN, SYSS\$QIOW SYSS\$DCLEXH						
															.ENTRY	SMG\$CREATE_PASTEBOARD, Save R2,R3,R4,R5,R6,-, R7,R8,R9,RT0,R11				1595	
															SUBL2	#44, SP					
															CLRW	FS_LEN				1668	
				50											SUBB3	#1,(AP), DIFF				1694	
															CMPB	DIFF,#4					
															BLEQU	1\$					
															MOVL	#SMG\$_WRONUMARG, RO					
															RET						
															ADDL3	#1,PBD_L_COUNT, RO				1696	
				50	00000000'										CMPL	R0,#16					
															BLEQ	2\$					
															MOVL	#SMG\$_TOOMANPAS, RO					
															RET						
															FFC	ZERO, PBD_K_MAX_PB_BY_REF, PBD_V_PB_AVAIL,					
															PBID						
															BBSS	PBID, PBD_V_PB_AVAIL, 3\$					
															MOVW	#10, FS_LEN				1702	
															MOVAB	P.AAA, FS_ADDR				1703	
															CMPB	(AP), #2				1705	
															BLSSU	4\$					
															TSTL	8(AP)					
															BEQL	4\$					
															MOVL	OUT_DEVICE, RO				1708	
															JSB	LIB\$ANALYZE_SDESC_R2					
															MOVL	R0, STATUS					
															MOVL	R2, R11					
															MOVL	R1, FS_LEN					
															BLBC	STATUS, 5\$					
															PUSHAB	PBCB				1718	
															PUSHAB	TERM_TYPE					
															MOVZWL	FS_LEN, -(SP)				1720	
															PUSHL	FS_ADDR				1719	
															CALLS	#4, SMG\$\$\$SETUP_TERMINAL_TYPE					
															MOVL	R0, STATUS					
															BLBS	STATUS, 6\$				1724	



		50	08	AE	D0	00090	6\$:	MOVL	PBCB, R0	1738
				51	D4	00094		CLRL	R1	
	42	8F	58	A0	91	00096		CMPB	88(R0), #66	
				02	13	00098		BEQL	7\$	
				51	D6	0009D		INCL	R1	
				52	D4	0009F	7\$:	CLRL	R2	1739
			10	A0	95	000A1		TSTB	16(R0)	
				02	12	000A4		BNEQ	8\$	
		52		51	D6	000A6		INCL	R2	
				51	C8	000A8	8\$:	BISL2	R1, R2	
		05	10	A0	91	000AB		CLRL	R1	1740
				02	12	000AD		CMPB	16(R0), #5	
				51	D6	000B1		BNEQ	9\$	
				52	89	000B3		INCL	R1	
00D0	C0	53	51	52	89	000B5	9\$:	BISB3	R2, R1, R3	
		01	03	53	F0	000B9		INSV	R3, #3, #1, 208(R0)	
		03	00D0	03	E1	000C0		BBC	#3, 208(R0), 10\$	1755
				00AB	31	000C6		BRW	22\$	
		57	00000000	EF	D0	000C9	10\$:	MOVL	PBD_L_COUNT, R7	1757
		56		01	CE	000D0		MNEGL	#1, -1	
				0095	31	000D3	11\$:	BRW	20\$	
		55	00000000	EF46	D0	000D6	12\$:	MOVL	PBD_A_PBCB[1], SEARCH_PBCB	1765
				F3	13	000DE		BEQL	11\$	
		54	08	AE	D0	000E0		MOVL	PBCB, R4	1769
	12	A4	12	A5	B1	000E4		CMPW	18(SEARCH_PBCB), 18(R4)	
				E8	12	000E9		BNEQ	11\$	
12	A4	00	18	A5	2D	000EB		CMPC5	18(SEARCH_PBCB), 24(SEARCH_PBCB), #0, -	1775
				18	A4	000F3			18(R4), 24(R4)	
				74	12	000F5		BNEQ	20\$	
		00	00000000	EF	5A	000F7		BBCC	PBID, PBD_V_PB_AVAIL, 13\$	1788
				08	A4	000FF	13\$:	TSTL	8(R4)	1793
				08	0B	00102		BEQL	14\$	
				08	A4	00104		PUSHL	8(R4)	1796
		0000V	CF	01	FB	00107		CALLS	#1, SMG\$DEALLOCATE_WCB	
			2D	50	E9	0010C		BLBC	STATUS, 16\$	1795
				6C	A4	0010F	14\$:	TSTL	108(R4)	1803
				15	13	00112		BEQL	15\$	
				6C	A4	00114		PUSHAB	108(R4)	1807
	04	AE	70	A4	3C	00117		MOVZWL	112(R4), 4(SP)	1806
			04	AE	9F	0011C		PUSHAB	4(SP)	
		00000000G	00	02	FB	0011F		CALLS	#2, LIB\$FREE_VM	1807
			13	50	E9	00126		BLBC	STATUS, 16\$	
				08	AE	00129	15\$:	PUSHAB	PBCB	1814
		04	AE	014C	8F	0012C		MOVZWL	#332, 4(SP)	
				04	AE	00132		PUSHAB	4(SP)	
		00000000G	00	02	FB	00135		CALLS	#2, LIB\$FREE_VM	
			01	50	E8	0013C	16\$:	BLBS	STATUS, 17\$	
					04	0013F		RET		
	04	BC	14	A5	D0	00140	17\$:	MOVL	20(SEARCH_PBCB), @NEW_PBID	1823
		03		6C	91	00145		CMPB	(AP), #3	1829
				0A	1F	00148		BLSSU	18\$	
			0C	AC	D5	0014A		TSTL	12(AP)	
				05	13	0014D		BEQL	18\$	
	0C	BC	5F	A5	9A	0014F		MOVZBL	95(SEARCH_PBCB), @PB_ROWS	1830
		04		6C	91	00154	18\$:	CMPB	(AP), #4	1832
				0A	1F	00157		BLSSU	19\$	
			10	AC	D5	00159		TSTL	16(AP)	



					05	13	0015C	BEQL	19\$		
	10	BC	5A	A5	3C	0015E	MOVZWL	90(SEARCH_PBCB), @PB_COLS		1833	
		50	00000000G	8F	D0	00163	MOVL	#SMG\$_PASALREXI, R0		1835	
					04	0016A	RET				
	02		56		57	F2	0016B	A0BLSS	R7, 1, 21\$	1757	
					03	11	0016F	BRB	22\$		
					FF62	31	00171	BRW	12\$		
		57	08	AE	D0	00174	21\$:				
		14	A7	5A	D0	00178	22\$:	MOVL	PBCB, R7	1846	
				C7	9F	0017C	MOVL	PBID, 20(R7)			
		04	AE	00B0	58	3C	00180	PUSHAB	176(R7)	1856	
				04	AE	9F	00184	MOVZWL	FS_LEN, 4(SP)		
		00000000G	00	02	FB	00187	PUSHAB	4(SP)			
			59	50	D0	0018E	CALLS	#2, LIB\$GET_VM			
			5F	59	E9	00191	MOVL	R0, STATUS			
		00E4	C7	58	B0	00194	BLBC	STATUS, 24\$		1857	
00B0	D7		6B	58	28	00199	MOVW	FS_LEN, 228(R7)		1858	
	03	00D0	C7	03	E0	0019F	MOVC3	FS_LEN, (FS_ADDR), @176(R7)		1859	
				00D3	31	001A5	BBS	#3, 208(R7), 23\$		1867	
				00E8	C7	9F	001A8	BRW	27\$		
		04	AE	50	8F	9A	001AC	PUSHAB	232(R7)	1874	
				04	AE	9F	001B1	MOVZBL	#80, 4(SP)		
		00000000G	00	02	FB	001B4	PUSHAB	4(SP)			
			59	50	D0	001BB	CALLS	#2, LIB\$GET_VM			
			32	59	E9	001BE	MOVL	R0, STATUS			
				00EC	C7	9F	001C1	BLBC	STATUS, 24\$	1875	
		04	AE	44	8F	9A	001C5	PUSHAB	236(R7)	1877	
				04	AE	9F	001CA	MOVZBL	#68, 4(SP)		
		00000000G	00	02	FB	001CD	PUSHAB	4(SP)			
			59	50	D0	001D4	CALLS	#2, LIB\$GET_VM			
			19	59	E9	001D7	MOVL	R0, STATUS			
				00F0	C7	9F	001DA	BLBC	STATUS, 24\$	1878	
		04	AE	5A	A7	3C	001DE	PUSHAB	240(R7)	1887	
				04	AE	D6	001E3	MOVZWL	90(R7), 4(SP)		
				04	AE	9F	001E6	INCL	4(SP)		
		00000000G	00	02	FB	001E9	PUSHAB	4(SP)			
			59	50	D0	001F0	CALLS	#2, LIB\$GET_VM			
			70	59	E9	001F3	MOVL	R0, STATUS			
			56	00E8	C7	D0	001F6	BLBC	STATUS, 25\$	1888	
0050	8F	00	6E	00	2C	001FB	MOVL	232(R7), R6		1904	
					66	00202	MOVC5	#0, (SP), #0, #80, (R6)			
			66	5003	8F	B0	00203	MOVW	#20483, (R6)		
		04	A6	40	8F	9A	00208	MOVZBL	#64, 4(R6)		
		16	A6		01	90	0020D	MOVB	#1, 22(R6)		
		18	A6		57	D0	00211	MOVL	R7, 24(R6)		
		1D	A6	0200	8F	B0	00215	MOVW	#512, 29(R6)		
		1F	A6		02	90	0021B	MOVB	#2, 31(R6)		
		2C	A6		5B	D0	0021F	MOVL	FS_ADDR, 44(R6)		
		30	A6	FDCC	CF	9E	00223	MOVAB	P.AAB, 48(R6)		
		34	A6		58	90	00229	MOVB	FS_LEN, 52(R6)		
		35	A6		0D	90	0022D	MOVB	#13, 53(R6)		
		36	A7		01	A1	00231	ADDW3	#1, 90(R7), 54(R6)		
0044	8F	00	5A	00EC	C7	D0	00237	MOVL	236(R7), R8	1910	
			6E		00	2C	0023C	MOVC5	#0, (SP), #0, #68, (R8)		
					68	00243					
			68	4401	8F	B0	00244	MOVW	#17409, (R8)		
		18	A8		57	D0	00249	MOVL	R7, 24(R8)		



28	A8	1E	A8	94	0024D	CLRB	30(R8)	
3C	A8	00F0	C7	D0	00250	MOVL	240(R7), 40(R8)	
			56	D0	00256	MOVL	R6, 60(R8)	
00000000G	00		56	DD	0025A	PUSHL	R6	1916
	59		01	FB	0025C	CALLS	#1, SYSS\$CREATE	
	0F		50	D0	00263	MOVL	R0, STATUS	1917
			59	E9	00266	BLBC	STATUS, 26\$	1923
00000000G	00		58	DD	00269	PUSHL	R8	
	59		01	FB	0026B	CALLS	#1, SYSS\$CONNECT	
	6E		50	D0	00272	MOVL	R0, STATUS	
			59	E8	00275	BLBS	STATUS, 29\$	1924
24	AE	12	00E1	31	00278	BRW	31\$	
28	AE	18	A7	3C	0027B	MOVZWL	18(R7), NAME_DESC	1939
			A7	9E	00280	MOVAB	24(R7), NAME_DESC+4	1940
			7E	7C	00285	CLRQ	-(SP)	1948
		64	A7	9F	00287	PUSHAB	100(R7)	
		30	AE	9F	0028A	PUSHAB	NAME_DESC	
00000000G	00		04	FB	0028D	CALLS	#4, SYSS\$ASSIGN	
	59		50	D0	00294	MOVL	R0, STATUS	
	DE		59	E9	00297	BLBC	STATUS, 26\$	1949
		0C	AE	9F	0029A	PUSHAB	ASYNCFN	1955
00000000G	00		01	FB	0029D	CALLS	#1, LIB\$GET_EF	
	59		50	D0	002A4	MOVL	R0, STATUS	
	CE		59	E9	002A7	BLBC	STATUS, 26\$	1956
67	A7	0C	AE	90	002AA	MOVB	ASYNCFN, 103(R7)	1962
			7E	7C	002AF	CLRQ	-(SP)	1976
			7E	7C	002B1	CLRQ	-(SP)	
			0C	DD	002B3	PUSHL	#12	
		24	AE	9F	002B5	PUSHAB	CHARBUF	
			7E	7C	002B8	CLRQ	-(SP)	
		3C	AE	9F	002BA	PUSHAB	TTIOSB	
			27	DD	002BD	PUSHL	#39	
	7E	64	A7	3C	002BF	MOVZWL	100(R7), -(SP)	
			7E	D4	002C3	CLRL	-(SP)	
00000000G	00		0C	FB	002C5	CALLS	#12, SYSS\$QIOW	
	59		50	D0	002CC	MOVL	R0, STATUS	
	A6		59	E9	002CF	BLBC	STATUS, 26\$	1977
	05	1C	AE	E8	002D2	BLBS	TTIOSB, 28\$	1978
	50	1C	AE	3C	002D6	MOVZWL	TTIOSB, R0	
			04	002DA	RET			
00B4	C7	1E	AE	D0	002DB	MOVL	TTIOSB+2, 180(R7)	1980
11	A7	22	AE	90	002E1	MOVB	TTIOSB+6, 17(R7)	1982
78	A7	00000000G	00	9E	002E6	MOVAB	SMG\$PBCB_EXIT_HANDLER, 120(R7)	1995
7C	A7		02	90	002EE	MOVB	#2, 124(R7)	1997
0080	C7	0088	C7	9E	002F2	MOVAB	136(R7), 128(R7)	1999
0084	C7		57	D0	002F9	MOVL	R7, 132(R7)	2005
		74	A7	9F	002FE	PUSHAB	116(R7)	2016
00000000G	00		01	FB	00301	CALLS	#1, SYSS\$DCLEXH	
	59		50	D0	00308	MOVL	R0, STATUS	
	4E		59	E9	0030B	BLBC	STATUS, 31\$	2017
008C	C7	6CFA0000	8F	D0	0030E	MOVL	#1828323328, 140(R7)	2048
0090	C7		8F	90	00317	MOVB	#-97, 144(R7)	2051
0091	C7	00000000G	00	9E	0031D	MOVAB	SMG\$OUT_OF_BAND_HANDLER, 145(R7)	2052
0095	C7		04	90	00326	MOVB	#4, 149(R7)	2053
		00000000'	EF	D6	0032B	INCL	PBD_L COUNT	2059
00000000'EF4A			57	D0	00331	MOVL	R7, -PBD_A_PBCB[PBD]	2061
50			01	D0	00339	MOVL	#1, CLEAR_FLAG	2067



		05		6C	91	0033C	CMPB	(AP), #5	2068
				09	1F	0033F	BLSSU	30\$	
			14	AC	D5	00341	TSTL	20(AP)	
				04	13	00344	BEQL	30\$	
		50		BC	D2	00346	MCOML	@PRESERVE_SCREEN_FLAG, CLEAR_FLAG	2069
		13		50	E9	0034A	BLBC	CLEAR_FLAG, 32\$	2071
				57	DD	0034D	PUSHL	R7	2074
	00000000G	00		01	FB	0034F	CALLS	#1, SMG\$ERASE_PASTEBOARD	
		59		50	D0	00356	MOVL	R0, STATUS	
		54		59	E8	00359	BLBS	STATUS, 35\$	2075
		50		59	D0	0035C	MOVL	STATUS, R0	
					04	0035F	RET		
		59	08	A7	D0	00360	MOVL	8(R7), WCB	2080
		56	28	A9	D0	00364	MOVL	40(WCB), R6	2082
56	20	6E		00	2C	00368	MOVCS	#0, (SP), #32, R6, @8(WCB)	
			08	B9		0036D			
56	20	6E		00	2C	0036F	MOVCS	#0, (SP), #32, R6, @20(WCB)	2083
			14	B9		00374			
56	00	6E		00	2C	00376	MOVCS	#0, (SP), #0, R6, @12(WCB)	2084
			0C	B9		0037B			
56	00	6E		00	2C	0037D	MOVCS	#0, (SP), #0, R6, @24(WCB)	2085
			18	B9		00382			
			10	A9	D5	00384	TSTL	16(WCB)	2087
				07	13	00387	BEQL	33\$	
56	00	6E		00	2C	00389	MOVCS	#0, (SP), #0, R6, @16(WCB)	2090
			10	B9		0038E			
			1C	A9	D5	00390	TSTL	28(WCB)	2093
				07	13	00393	BEQL	34\$	
56	00	6E		00	2C	00395	MOVCS	#0, (SP), #0, R6, @28(WCB)	2096
			1C	B9		0039A			
		56	02	A9	3C	0039C	MOVZWL	2(WCB), R6	2112
				56	D6	003A0	INCL	R6	
56	00	6E		00	2C	003A2	MOVCS	#0, (SP), #0, R6, @44(WCB)	
			2C	B9		003A7			
56	00	6E		00	2C	003A9	MOVCS	#0, (SP), #0, R6, @48(WCB)	2113
			30	B9		003AE			
		03		6C	91	003B0	CMPB	(AP), #3	2121
				0A	1F	003B3	BLSSU	36\$	
			0C	AC	D5	003B5	TSTL	12(AP)	
				05	13	003B8	BEQL	36\$	
	0C	BC	5F	A7	9A	003BA	MOVZBL	95(R7), @PB_ROWS	
		04		6C	91	003BF	CMPB	(AP), #4	2122
				0A	1F	003C2	BLSSU	37\$	
			10	AC	D5	003C4	TSTL	16(AP)	
				05	13	003C7	BEQL	37\$	
	10	BC	5A	A7	3C	003C9	MOVZWL	90(R7), @PB_COLS	
	04	BC		5A	D0	003CE	MOVL	PBID, @NEW_PBID	2127
		50		01	D0	003D2	MOVL	#1, R0	2129
					04	003D5	RET		2130

; Routine Size: 982 bytes, Routine Base: \_SMG\$CODE + 06BF

; 1879 2131 1 !<BLF/PAGE>



```

: 1881      2132 1 %SBTTL 'SMG$DELETE_PASTEBOARD - Delete Pasteboard'
: 1882      2133 1 GLOBAL ROUTINE SMG$DELETE_PASTEBOARD ( PBID, CLEAR_SCREEN_FLAG ) =
: 1883      2134 1
: 1884      2135 1 !++
: 1885      2136 1 FUNCTIONAL DESCRIPTION:
: 1886      2137 1
: 1887      2138 1     This routine terminates all use of a given physical display.
: 1888      2139 1     It deallocates the pasteboard control block and all its
: 1889      2140 1     substructures. It gets rid of the event flag and the channel
: 1890      2141 1     number. It removes any associated exit handler.
: 1891      2142 1
: 1892      2143 1 CALLING SEQUENCE:
: 1893      2144 1
: 1894      2145 1     ret_status.wlc.v = SMG$DELETE_PASTEBOARD ( PBID.rl.r
: 1895      2146 1     [,CLEAR_SCREEN_FLAG.rl.r])
: 1896      2147 1
: 1897      2148 1 FORMAL PARAMETERS:
: 1898      2149 1
: 1899      2150 1     PBID.rl.r           Pasteboard id of pasteboard.
: 1900      2151 1
: 1901      2152 1     CLEAR_SCREEN_FLAG.rl.r Set to 1 to clear the screen,
: 1902      2153 1     0 to keep it as is.
: 1903      2154 1     The default is to clear the screen.
: 1904      2155 1
: 1905      2156 1 IMPLICIT INPUTS:
: 1906      2157 1
: 1907      2158 1     NONE
: 1908      2159 1
: 1909      2160 1 IMPLICIT OUTPUTS:
: 1910      2161 1
: 1911      2162 1     NONE
: 1912      2163 1
: 1913      2164 1 COMPLETION STATUS:
: 1914      2165 1
: 1915      2166 1     SS$ NORMAL      Normal successful completion
: 1916      2167 1     SMG$_WRONUMARG  Wrong number of arguments.
: 1917      2168 1     SS$_xyz         errors from $DASSGN
: 1918      2169 1     LIB$_xyz        errors from LIB$FREE_VM or LIB$FREE_EF
: 1919      2170 1     SMG$_xyz        errors from SMG$$FLUSH_BUFFER
: 1920      2171 1
: 1921      2172 1 SIDE EFFECTS:
: 1922      2173 1
: 1923      2174 1     NONE
: 1924      2175 1 !--

```



```

: 1926      2176 2 BEGIN
: 1927      2177 2
: 1928      2178 2 BUILTIN
: 1929      2179 2     NULLPARAMETER;
: 1930      2180 2
: 1931      2181 2 LOCAL
: 1932      2182 2
: 1933      2183 2     STATUS,                ! Status of subroutine calls
: 1934      2184 2     CURR_PP : REF $PP_DECL,    ! Pasting packet pointer
: 1935      2185 2     WCB      : REF $WCB_DECL,   ! Window control block.
: 1936      2186 2     PBCB     : REF $PBCB_DECL;  ! Address of pasteboard control
: 1937      2187 2                                     ! block
: 1938      2188 2
: 1939      2189 2
: 1940      2190 2
: 1941      2191 2
: 1942      2192 2 EXTERNAL ROUTINE
: 1943      2193 2
: 1944      2194 2     SMG$$FORCE_SCROLL_REG,
: 1945      2195 2     SMG$$ERASE_PASTEBOARD,
: 1946      2196 2     SMG$$FLUSH_BUFFER,
: 1947      2197 2     SMG$CHANGE_PBD_CHARACTERISTICS;
: 1948      2198 2
: 1949      2199 2 $SMG$VALIDATE_ARGCOUNT (1, 2); ! Test for right no. of args
: 1950      2200 2
: 1951      2201 2 $SMG$GET_PBCB (.PBID,PBCB);      ! Get address of PBCB
: 1952      2202 2
: 1953      2203 2 !+
: 1954      2204 2 ! Batch up the unpastes, so that the whole screen disappears at once.
: 1955      2205 2 !-
: 1956      2206 2     IF NOT (STATUS = SMG$$BEGIN_PASTEBOARD_UPDATE_R1(.PBCB))
: 1957      2207 2     THEN
: 1958      2208 2         RETURN (.STATUS);
: 1959      2209 2
: 1960      2210 2 !+
: 1961      2211 2 ! Walk chain of all DCB's pasted to this pasteboard and unpaste each.
: 1962      2212 2 !-
: 1963      2213 2     CURR_PP = .PBCB [PBCB_A_PP_PREV];
: 1964      2214 2     WHILE .CURR_PP NEQ PBCB [PBCB_A_PP_NEXT]
: 1965      2215 2     DO
: 1966      2216 2         BEGIN ! Walk chain
: 1967      2217 2         LOCAL
: 1968      2218 2             DCB      : REF $DCB_DECL,    ! Address of DCB involved
: 1969      2219 2             PP_BASE   : REF $PP_DECL;    ! Base addr of this PP
: 1970      2220 2
: 1971      2221 2             PP_BASE = .CURR_PP - PP_PBCB_QUEUE_OFFSET; ! Since queue header
: 1972      2222 2                                     ! not at top of
: 1973      2223 2                                     ! structure.
: 1974      2224 2             DCB = .PP_BASE [PP_A_DCB_ADDR];
: 1975      2225 2             IF NOT (STATUS = SMG$$UNPASTE_VIRTUAL_DISPLAY (
: 1976      2226 2                 :DCB,                ! DCB involved
: 1977      2227 2                 :PBCB))                ! PBCB involved
: 1978      2228 2             THEN
: 1979      2229 2                 RETURN (.STATUS);
: 1980      2230 2
: 1981      2231 2             CURR_PP = .PP_BASE [PP_A_PREV_PBCB]; ! Step to next PP
: 1982      2232 2         END; ! Walk chain

```

B  
C  
D  
E  
F  
G  
H  
I  
J  
K  
L  
M  
N  
O  
P  
Q  
R  
S  
T  
U  
V  
W  
X  
Y  
Z



```

: 1983      2233      2
: 1984      2234      2      PBCB[PBCB_L_BATCH_LEVEL]=0;
: 1985      2235      2
: 1986      2236      2
: 1987      2237      2      !+
: 1988      2238      2      If the user asked for the screen to be erased, then
: 1989      2239      2      release lock on pasteboard, force output of now-blank screen, and
: 1990      2240      2      flush it out.
: 1991      2241      2      !-
: 1992      2242      3      IF NULLPARAMETER(CLEAR_SCREEN_FLAG)
: 1993      2243      2      OR (NOT NULLPARAMETER(CLEAR_SCREEN_FLAG) AND ..CLEAR_SCREEN_FLAG)
: 1994      2244      2      THEN
: 1995      2245      3      BEGIN      ! clear screen
: 1996      2246      3      ! (b)      IF NOT (STATUS = SMG$$END_PASTEBOARD_UPDATE_R2(.PBCB))
: 1997      2247      3      ! (b)      THEN
: 1998      2248      3      ! (b)      RETURN (.STATUS);
: 1999      2249      3
: 2000      2250      3      ! (b)      IF NOT ( STATUS = SMG$$CHECK_FOR_OUTPUT_PBCB(.PBCB))
: 2001      2251      3      ! (b)      THEN
: 2002      2252      3      ! (b)      RETURN (.STATUS);
: 2003      2253      3
: 2004      2254      3      ! (b)      IF NOT (STATUS = SMG$$FLUSH_BUFFER(.PBCB))
: 2005      2255      3      ! (b)      THEN
: 2006      2256      3      ! (b)      RETURN (.STATUS);
: 2007      2257      3
: 2008      2258      3      ! Note (b): Erase pasteboard should clear the screen and
: 2009      2259      3      we can bypass flushing since the user is deleting his
: 2010      2260      3      pasteboard anyhow.
: 2011      2261      3
: 2012      2262      3      PBCB[PBCB_V_BUF_ENABLED]=0;
: 2013      2263      3
: 2014      2264      4      IF NOT (STATUS = SMG$$ERASE_PASTEBOARD(.PBCB))
: 2015      2265      3      THEN
: 2016      2266      3      RETURN (.STATUS);
: 2017      2267      3
: 2018      2268      3      !+
: 2019      2269      3      ! Set terminal back to it's original width.
: 2020      2270      3      ! This requires batching to be off.
: 2021      2271      3      !-
: 2022      2272      3
: 2023      2273      3      IF .PBCB[PBCB_W_WIDTH] NEQ .PBCB[PBCB_W_ORIG_WIDTH]
: 2024      2274      4      THEN BEGIN
: 2025      2275      4      STATUS=SMG$CHANGE_PBD_CHARACTERISTICS(.PBCB,
: 2026      2276      4      %REF(.PBCB[PBCB_W_ORIG_WIDTH]));
: 2027      2277      4      IF NOT .STATUS THEN RETURN .STATUS;
: 2028      2278      4      STATUS=SMG$$FLUSH_BUFFER(.PBCB);
: 2029      2279      4      IF NOT .STATUS THEN RETURN .STATUS
: 2030      2280      3      END;
: 2031      2281      3
: 2032      2282      3      END      ! clear screen
: 2033      2283      2      ELSE
: 2034      2284      3      BEGIN
: 2035      2285      3      SMG$$FLUSH_BUFFER(.PBCB);
: 2036      2286      3      PBCB[PBCB_V_BUF_ENABLED]=0;
: 2037      2287      2      END;
: 2038      2288      2
: 2039      2289      2      WCB=.PBCB[PBCB_A_WCB];

```



```

: 2040      2290  2
: 2041      2291  2
: 2042      2292  2  !+
: 2043      2293  2  ! If a scrolling region is set (other than the full screen),
: 2044      2294  2  ! then reset it now, being careful to leave the cursor alone
: 2045      2295  2  ! even though SET SCROLLING REGION may move it.
: 2046      2296  2  ! Note that if we never established any scrolling regions,
: 2047      2297  2  ! the TOP_SCROLL line will be 0.
: 2048      2298  2  !-
: 2049      2299  2  IF .PBCB[PBCB_W_TOP_SCROLL_LINE] NEQ 0
: 2050      2300  3  AND (.PBCB[PBCB_W_TOP_SCROLL_LINE] NEQ 1 OR
: 2051      2301  3  .PBCB[PBCB_W_BOT_SCROLL_LINE] NEQ .WCB[WCB_W_NO_ROWS])
: 2052      2302  2  THEN
: 2053      2303  3  BEGIN      ! Remove scrolling regions
: 2054      2304  3
: 2055      2305  3  LOCAL
: 2056      2306  3
: 2057      2307  3  FINAL_ROW,      ! Final cursor row
: 2058      2308  3  FINAL_COL:      ! Final cursor column
: 2059      2309  3
: 2060      2310  3  !+
: 2061      2311  3  ! Construct escape sequence (possibly null if not a supporting terminal)
: 2062      2312  3  ! to set the hardware scroll region to the full height of the screen.
: 2063      2313  3  !-
: 2064      2314  3
: 2065      2315  3  $SMG$GET_TERM_DATA(SET_SCROLL_REGION,
: 2066      2316  3  1,
: 2067      2317  3  .WCB[WCB_W_NO_ROWS]);
: 2068      2318  3
: 2069      2319  3  !+
: 2070      2320  3  ! Output BUFFER.
: 2071      2321  3  !-
: 2072      2322  3
: 2073      2323  3  IF .PBCB[PBCB_L_CAP_LENGTH] NEQ 0
: 2074      2324  3  THEN
: 2075      2325  4  BEGIN      ! Issue the reset
: 2076      2326  4
: 2077      2327  4  !+
: 2078      2328  4  ! Remember where the user left the physical cursor, since
: 2079      2329  4  ! changing scrolling regions might upset this.
: 2080      2330  4  !-
: 2081      2331  4
: 2082      2332  4  FINAL_ROW=.WCB[WCB_W_CURR_CUR_ROW];
: 2083      2333  4  FINAL_COL=.WCB[WCB_W_CURR_CUR_COL];
: 2084      2334  4
: 2085      2335  4  STATUS = SMG$$OUTPUT(.PBCB,.PBCB[PBCB_L_CAP_LENGTH],
: 2086      2336  4  .PBCB[PBCB_A_CAP_BUFFER]);
: 2087      2337  4  IF NOT .STATUS THEN RETURN .STATUS;
: 2088      2338  4
: 2089      2339  4  !+
: 2090      2340  4  ! Move the cursor back to where it was.
: 2091      2341  4  !-
: 2092      2342  4
: 2093      2343  4  IF NOT NULLPARAMETER(CLEAR_SCREEN_FLAG)
: 2094      2344  4  AND NOT ..CLEAR_SCREEN_FLAG
: 2095      2345  5  THEN BEGIN      ! Restore final cursor position
: 2096      2346  5

```



```

: 2097      2347 5      $SMG$GET_TERM_DATA(SET_CURSOR_ABS,.FINAL_ROW,.FINAL_COL);
: 2098      2348 5
: 2099      2349 5      STATUS = SMG$$OUTPUT(.PBCB,.PBCB[PBCB_L_CAP_LENGTH],
: 2100      2350 5      .PBCB[PBCB_A_CAP_BUFFER]);
: 2101      2351 5      IF NOT .STATUS THEN RETURN .STATUS
: 2102      2352 5
: 2103      2353 5      END      ! Restore final cursor position
: 2104      2354 5
: 2105      2355 4      END      ! Issue the reset
: 2106      2356 4
: 2107      2357 2      END;      ! Remove scrolling regions
: 2108      2358 2
: 2109      2359 2      !+
: 2110      2360 2      ! Get rid of our exit handler. Ignore a no handler found error.
: 2111      2361 2      !-
: 2112      2362 2
: 2113      2363 2      STATUS=$CANEXH(DESBK=PBCB[PBCB_R_EXIT_BLOCK]);
: 2114      2364 3      IF (NOT .STATUS) AND (.STATUS NEQ-SS$_NOHANDLER)
: 2115      2365 2      THEN RETURN .STATUS;
: 2116      2366 2
: 2117      2367 2      !+
: 2118      2368 2      ! Deallocate the WCB if there is one.
: 2119      2369 2      !-
: 2120      2370 2
: 2121      2371 2      IF .PBCB[PBCB_A_WCB] NEQ 0
: 2122      2372 3      THEN BEGIN      ! getting rid of WCB
: 2123      2373 3
: 2124      2374 3      STATUS=$DEALLOCATE_WCB(.PBCB[PBCB_A_WCB]);
: 2125      2375 3      PBCB[PBCB_A_WCB]=0;      ! safety
: 2126      2376 3      IF NOT .STATUS THEN RETURN .STATUS
: 2127      2377 3
: 2128      2378 2      END;      ! getting rid of WCB
: 2129      2379 2
: 2130      2380 2      !+
: 2131      2381 2      ! If there is a channel assigned, deassign it now.
: 2132      2382 2      ! This automatically cancels any I/O on the channel.
: 2133      2383 2      ! In particular, it removes any out-of-band ASTs that
: 2134      2384 2      ! were enabled.
: 2135      2385 2      !-
: 2136      2386 2
: 2137      2387 2      IF .PBCB[PBCB_W_CHAN] NEQ 0
: 2138      2388 3      THEN BEGIN      ! deassigning channel
: 2139      2389 3
: 2140      2390 3      STATUS=$DASSGN(CHAN=.PBCB[PBCB_W_CHAN]);
: 2141      2391 3      PBCB[PBCB_W_CHAN]=0;      ! just in case we get called
: 2142      2392 3      ! again after returning an error
: 2143      2393 3      IF NOT .STATUS THEN RETURN .STATUS
: 2144      2394 3
: 2145      2395 2      END;      ! deassigning channel
: 2146      2396 2
: 2147      2397 2      !+
: 2148      2398 2      ! Free the event flags now.
: 2149      2399 2      ! Ignore error if it was already free.
: 2150      2400 2      !-
: 2151      2401 2
: 2152      2402 2      IF .PBCB[PBCB_B_EFN] NEQ 0
: 2153      2403 3      THEN BEGIN

```



```

: 2154      2404 3      STATUS=LIB$FREE_EF( %REF(.PBCB[PBCB_B_EFN]) );
: 2155      2405 4      IF (NOT .STATUS) AND (.STATUS NEQ LIB$_EF_ALRFRE)
: 2156      2406 3      THEN RETURN .STATUS;
: 2157      2407 3      PBCB[PBCB_B_EFN]=0
: 2158      2408 2      END;
: 2159      2409 2
: 2160      2410 2 IF .PBCB[PBCB_B_ASYNC_EFN] NEQ 0
: 2161      2411 3 THEN BEGIN
: 2162      2412 3     STATUS=LIB$FREE_EF( %REF(.PBCB[PBCB_B_ASYNC_EFN]) );
: 2163      2413 4     IF (NOT .STATUS) AND (.STATUS NEQ LIB$_EF_ALRFRE)
: 2164      2414 3     THEN RETURN .STATUS;
: 2165      2415 3     PBCB[PBCB_B_ASYNC_EFN]=0
: 2166      2416 2     END;
: 2167      2417 2
: 2168      2418 2 !+
: 2169      2419 2 ! Free the output buffer now.
: 2170      2420 2 !-
: 2171      2421 2
: 2172      2422 2 IF .PBCB[PBCB_A_OUTPUT_BUFFER] NEQ 0
: 2173      2423 3 THEN BEGIN ! freeing output buffer
: 2174      2424 3
: 2175      2425 3     STATUS=LIB$FREE_VM(%REF(.PBCB[PBCB_W_OUTPUT_BUFSIZ] ),
: 2176      2426 3     PBCB[PBCB_A_OUTPUT_BUFFER] );
: 2177      2427 3     PBCB[PBCB_A_OUTPUT_BUFFER]=0;
: 2178      2428 3     IF NOT .STATUS THEN RETURN .STATUS
: 2179      2429 3
: 2180      2430 2     END; ! freeing output buffer
: 2181      2431 2
: 2182      2432 2 !+
: 2183      2433 2 ! Free the output filename.
: 2184      2434 2 !-
: 2185      2435 2
: 2186      2436 2 IF .PBCB[PBCB_W_OUTNAM_LEN] NEQ 0
: 2187      2437 3 THEN BEGIN ! freeing outname
: 2188      2438 3     STATUS=LIB$FREE_VM(%REF(.PBCB[PBCB_W_OUTNAM_LEN] ),
: 2189      2439 3     PBCB[PBCB_A_OUTNAM] );
: 2190      2440 3     PBCB[PBCB_W_OUTNAM_LEN]=0;
: 2191      2441 3     IF NOT .STATUS THEN RETURN .STATUS
: 2192      2442 2     END; ! freeing outname
: 2193      2443 2
: 2194      2444 2 !+
: 2195      2445 2 ! Close the output file, if there was one.
: 2196      2446 2 !-
: 2197      2447 2
: 2198      2448 2 IF .PBCB[PBCB_A_FAB] NEQ 0
: 2199      2449 3 THEN BEGIN ! Close output file
: 2200      2450 3     STATUS=$CLOSE( FAB = .PBCB[PBCB_A_FAB] );
: 2201      2451 3     IF NOT .STATUS THEN RETURN .STATUS
: 2202      2452 2     END; ! Close output file
: 2203      2453 2
: 2204      2454 2 !+
: 2205      2455 2 ! Free the record buffer, if there was one.
: 2206      2456 2 !-
: 2207      2457 2
: 2208      2458 2 IF .PBCB[PBCB_A_RBF] NEQ 0
: 2209      2459 3 THEN BEGIN
: 2210      2460 3     STATUS=LIB$GET_VM(%REF(.PBCB[PBCB_W_WIDTH]+1),PBCB[PBCB_A_RBF]);

```



```

: 2211      2461      3      PBCB[PBCB_A_RBF]=0;
: 2212      2462      3      IF NOT .STATUS THEN RETURN .STATUS;
: 2213      2463      2      END;
: 2214      2464      2
: 2215      2465      2      !+
: 2216      2466      2      ! Free any FAB or RAB that was created.
: 2217      2467      2      !-
: 2218      2468      2
: 2219      2469      2      IF .PBCB[PBCB_A_FAB] NEQ 0
: 2220      2470      3      THEN BEGIN ! freeing FAB
: 2221      2471      3          STATUS=LIB$FREE_VM(%REF (FAB$C_BLN),
: 2222      2472      3              PBCB[PBCB_A_FAB]);
: 2223      2473      3          PBCB[PBCB_A_FAB]=0;
: 2224      2474      3          IF NOT .STATUS THEN RETURN .STATUS
: 2225      2475      2          END; ! freeing FAB
: 2226      2476      2
: 2227      2477      2      IF .PBCB[PBCB_A_RAB] NEQ 0
: 2228      2478      3      THEN BEGIN ! freeing RAB
: 2229      2479      3          STATUS=LIB$FREE_VM(%REF (RAB$C_BLN),
: 2230      2480      3              PBCB[PBCB_A_RAB]);
: 2231      2481      3          PBCB[PBCB_A_RAB]=0;
: 2232      2482      3          IF NOT .STATUS THEN RETURN .STATUS
: 2233      2483      2          END; ! freeing RAB
: 2234      2484      2
: 2235      2485      2      !+
: 2236      2486      2      ! Now go free the PBCB itself.
: 2237      2487      2      !-
: 2238      2488      2
: 2239      2489      3      IF NOT (STATUS=LIB$FREE_VM (%REF (PBCB_K_SIZE), PBCB))
: 2240      2490      2      THEN
: 2241      2491      2          RETURN (.STATUS);
: 2242      2492      2
: 2243      2493      2      !+
: 2244      2494      2      ! Since all went well, we can now adjust the count of how many PBCB's
: 2245      2495      2      ! we have and remove its address from the pasteboard directory.
: 2246      2496      2      !-
: 2247      2497      2
: 2248      2498      2      PBD_V_PB_AVAIL [..PBID] = 0;
: 2249      2499      2
: 2250      2500      2      PBD_L_COUNT = .PBD_L_COUNT - 1;
: 2251      2501      2
: 2252      2502      2      PBD_A_PBCB [..PBID] = 0;
: 2253      2503      2
: 2254      2504      2      RETURN SS$_NORMAL
: 2255      2505      2
: 2256      2506      1      END;

```

! Routine SMG\$DELETE\_PASTEBOARD

.EXTRN SYSSCANEXH, SYSSDASSGN  
.EXTRN SYSSCLOSE

.ENTRY SMG\$DELETE\_PASTEBOARD, Save R2,R3,R4,R5,R6,-; 2133  
R7,R8,R9,R10,R11  
MOVAB LIB\$FREE\_VM, R11  
MOVAB PBD\_L\_COUNT, R10  
SUBL2 #20, SP

OFFC 00000

5B 00000000G 00 9E 00002  
5A 00000000' EF 9E 00009  
5E 14 C2 00010



50	6C	01	83	00013	SUBB3	#1, (AP), DIFF	2199
	01	50	91	00017	CMPB	DIFF, #1	
		08	1B	0001A	BLEQU	1\$	
	50	00000000G	8F	D0 0001C	MOVL	#SMG\$_WRONUMARG, R0	
			04	00023	RET		
	59	04	AC	D0 00024	MOVL	PBID, R9	2201
	50		69	D0 00028	MOVL	(R9), R0	
			0A	19 0002B	BLSS	2\$	
	6A		50	D1 0002D	CMPL	R0, PBD_L_COUNT	
			05	14 00030	BGTR	2\$	
08	44	AA	50	E0 00032	BBS	R0, PBD V PB_AVAIL, 3\$	
	50	00000000G	8F	D0 00037	MOVL	#SMG\$_INVPAS_ID, R0	
			04	0003E	RET		
	04	AE	04	AA40 D0 0003F	MOVL	PBD_A_PBCB[R0], PBCB	
		54	04	AE D0 00045	MOVL	PBCB, R4	2206
		50		54 D0 00049	MOVL	R4, R0	
		00000000G	00	16 0004C	JSB	SMG\$\$BEGIN_PASTEBOARD_UPDATE_R1	
	55		50	D0 00052	MOVL	R0, STATUS	
	6E		55	E9 00055	BLBC	STATUS, 7\$	
	53	04	A4	D0 00058	MOVL	4(R4), CURR_PP	2213
	54		53	D1 0005C	CMPL	CURR_PP, R4	2214
			1B	13 0005F	BEQL	5\$	
	52	F8	A3	9E 00061	MOVAB	-8(R3), PP_BASE	2221
	50	10	A2	D0 00065	MOVL	16(PP_BASE), DCB	2224
			11	BB 00069	PUSHR	#*M<R0, R4>	2226
0000V	CF		02	FB 0006B	CALLS	#2, SMG\$\$UNPASTE_VIRTUAL_DISPLAY	
	55		50	D0 00070	MOVL	R0, STATUS	
	62		55	E9 00073	BLBC	STATUS, 8\$	2225
	53	0C	A2	D0 00076	MOVL	12(PP_BASE), CURR_PP	2231
			E0	11 0007A	BRB	4\$	2214
		00A4	C4	D4 0007C	CLRL	164(R4)	2234
	02		6C	91 00080	CMPB	(AP), #2	2241
			13	1F 00083	BLSSU	6\$	
		08	AC	D5 00085	TSTL	8(AP)	
			0E	13 00088	BEQL	6\$	
	02		6C	91 0008A	CMPB	(AP), #2	2242
			4C	1F 0008D	BLSSU	9\$	
		08	AC	D5 0008F	TSTL	8(AP)	
			47	13 00092	BEQL	9\$	
	43	08	BC	E9 00094	BLBC	@CLEAR_SCREEN_FLAG, 9\$	
0C	A4		01	8A 00098	BICB2	#1, 12(R4)	2262
			54	DD 0009C	PUSHL	R4	2264
00000000G	00		01	FB 0009E	CALLS	#1, SMG\$\$ERASE_PASTEBOARD	
	55		50	D0 000A5	MOVL	R0, STATUS	
	2D		55	E9 000A8	BLBC	STATUS, 8\$	
00E6	C4	5A	A4	B1 000AB	CMPW	90(R4), 230(R4)	2273
			35	13 000B1	BEQL	10\$	
	6E	00E6	C4	3C 000B3	MOVZWL	230(R4), (SP)	2276
		4200	8F	BB 000B8	PUSHR	#*M<R9, SP>	2275
00000000G	00		02	FB 000BC	CALLS	#2, SMG\$CHANGE_PBD_CHARACTERISTICS	
	55		50	D0 000C3	MOVL	R0, STATUS	
	0F		55	E9 000C6	BLBC	STATUS, 8\$	2277
			54	DD 000C9	PUSHL	R4	2278
00000000G	00		01	FB 000CB	CALLS	#1, SMG\$\$FLUSH_BUFFER	
	55		50	D0 000D2	MOVL	R0, STATUS	
	10		55	E8 000D5	BLBS	STATUS, 10\$	2279
		023D	31	000D8	BRW	38\$	



00000000G	00		54	DD	000DB	9\$:	PUSHL	R4	:	2285
OC	A4		01	FB	000DD		CALLS	#1, SMG\$\$FLUSH_BUFFER	:	
	53	08	01	8A	000E4		BICB2	#1, 12(R4)	:	2286
	50	00F4	A4	D0	000E8	10\$:	MOVL	8(R4), WCB	:	2289
			C4	3C	000EC		MOVZWL	244(R4), R0	:	2299
			7A	13	000F1		BEQL	14\$	:	
	01		50	B1	000F3		CMPL	R0, #1	:	2300
			08	12	000F6		BNEQ	11\$	:	
02	A3	00F6	C4	B1	000F8		CMPL	246(R4), 2(WCB)	:	2301
			6D	13	000FE		BEQL	14\$	:	
	52	0108	C4	9E	00100	11\$:	MOVAB	264(R4), R2	:	2317
	56	00FC	C4	9E	00105		MOVAB	252(R4), R6	:	
			66	D5	0010A		TSTL	(R6)	:	
			04	12	0010C		BNEQ	12\$	:	
			62	D4	0010E		CLRL	(R2)	:	
			2F	11	00110		BRB	13\$	:	
08	AE		02	D0	00112	12\$:	MOVL	#2, INPUT_ARGS	:	
OC	AE		01	D0	00116		MOVL	#1, INPUT_ARGS+4	:	
10	AE	02	A3	3C	0011A		MOVZWL	2(WCB), INPUT_ARGS+8	:	
		08	AE	9F	0011F		PUSHAB	INPUT_ARGS	:	
		0104	C4	DD	00122		PUSHL	260(R4)	:	
			52	DD	00126		PUSHL	R2	:	
		0100	C4	9F	00128		PUSHAB	256(R4)	:	
10	AE	023C	8F	3C	0012C		MOVZWL	#572, 16(SP)	:	
		10	AE	9F	00132		PUSHAB	16(SP)	:	
			56	DD	00135		PUSHL	R6	:	
00000000G	00		06	FB	00137		CALLS	#6, SMG\$GET_TERM_DATA	:	
	63		50	E9	0013E		BLBC	STATUS, 16\$	:	
			62	D5	00141	13\$:	TSTL	(R2)	:	2323
			76	13	00143		BEQL	19\$	:	
	58	20	A3	32	00145		CVTL	32(WCB), FINAL_ROW	:	2332
	57	22	A3	32	00149		CVTL	34(WCB), FINAL_COL	:	2333
	53	0104	C4	9E	0014D		MOVAB	260(R4), R3	:	2336
			63	DD	00152		PUSHL	(R3)	:	
			62	DD	00154		PUSHL	(R2)	:	2335
			54	DD	00156		PUSHL	R4	:	
00000000G	00		03	FB	00158		CALLS	#3, SMG\$OUTPUT	:	
	55		50	D0	0015F		MOVL	R0, STATUS	:	
	53		55	E9	00162		BLBC	STATUS, 18\$	:	2337
	02		6C	91	00165		CMPL	(AP), #2	:	2343
			51	1F	00168		BLSSU	19\$	:	
		08	AC	D5	0016A		TSTL	8(AP)	:	
			4C	13	0016D	14\$:	BEQL	19\$	:	
	48	08	BC	E8	0016F		BLBS	@CLEAR_SCREEN_FLAG, 19\$	:	2344
			66	D5	00173		TSTL	(R6)	:	2347
			04	12	00175		BNEQ	15\$	:	
			62	D4	00177		CLRL	(R2)	:	
			2D	11	00179		BRB	17\$	:	
08	AE		02	D0	0017B	15\$:	MOVL	#2, INPUT_ARGS	:	
OC	AE		58	D0	0017F		MOVL	FINAL_ROW, INPUT_ARGS+4	:	
10	AE		57	D0	00183		MOVL	FINAL_COL, INPUT_ARGS+8	:	
		08	AE	9F	00187		PUSHAB	INPUT_ARGS	:	
			63	DD	0018A		PUSHL	(R3)	:	
			52	DD	0018C		PUSHL	R2	:	
		0100	C4	9F	0018E		PUSHAB	256(R4)	:	
10	AE	023A	8F	3C	00192		MOVZWL	#570, 16(SP)	:	
		10	AE	9F	00198		PUSHAB	16(SP)	:	



00000000G	00	56	DD	0019B	PUSHL	R6		
	01	06	FB	0019D	CALLS	#6, SMG\$GET_TERM_DATA		
		50	E8	001A4	BLBS	STATUS, 17\$		
				04	RET			
		63	DD	001A8	PUSHL	(R3)		2350
		62	DD	001AA	PUSHL	(R2)		2349
		54	DD	001AC	PUSHL	R4		
00000000G	00	03	FB	001AE	CALLS	#3, SMG\$\$OUTPUT		
	55	50	D0	001B5	MOVL	R0, STATUS		
	48	55	E9	001B8	BLBC	STATUS, 22\$		2351
		74	A4	9F 001BB	PUSHAB	116(R4)		2363
00000000G	00	01	FB	001BE	CALLS	#1, SYS\$CANEXH		
	55	50	D0	001C5	MOVL	R0, STATUS		
	09	55	E8	001C8	BLBS	STATUS, 20\$		2364
000008F8	8F	55	D1	001CB	CMPL	STATUS, #2296		
		78	12	001D2	BNEQ	26\$		
		08	A4	D5 001D4	TSTL	8(R4)		2371
			14	13 001D7	BEQL	21\$		
		08	A4	DD 001D9	PUSHL	8(R4)		2374
0000V	CF	01	FB	001DC	CALLS	#1, SMG\$\$DEALLOCATE_WCB		
	55	50	D0	001E1	MOVL	R0, STATUS		
		08	A4	D4 001E4	CLRL	8(R4)		2375
	03	55	E8	001E7	BLBS	STATUS, 21\$		2376
		0080	31	001EA	BRW	30\$		
		64	A4	B5 001ED	TSTW	100(R4)		2387
			17	13 001F0	BEQL	23\$		
	7E	64	A4	3C 001F2	MOVZWL	100(R4), -(SP)		2390
00000000G	00	01	FB	001F6	CALLS	#1, SYS\$DASSGN		
	55	50	D0	001FD	MOVL	R0, STATUS		
		64	A4	B4 00200	CLRW	100(R4)		2391
	03	55	E8	00203	BLBS	STATUS, 23\$		2393
		0083	31	00206	BRW	32\$		
		66	A4	95 00209	TSTB	102(R4)		2402
			1F	13 0020C	BEQL	25\$		
	6E	66	A4	9A 0020E	MOVZBL	102(R4), (SP)		2404
			5E	DD 00212	PUSHL	SP		
00000000G	00	01	FB	00214	CALLS	#1, LIB\$FREE_EF		
	55	50	D0	0021B	MOVL	R0, STATUS		
	09	55	E8	0021E	BLBS	STATUS, 24\$		2405
00000000G	8F	55	D1	00221	CMPL	STATUS, #LIB\$_EF_ALRFRE		
		22	12	00228	BNEQ	26\$		
		66	A4	94 0022A	CLRB	102(R4)		2407
		67	A4	95 0022D	TSTB	103(R4)		2410
			22	13 00230	BEQL	29\$		
	6E	67	A4	9A 00232	MOVZBL	103(R4), (SP)		2412
			5E	DD 00236	PUSHL	SP		
00000000G	00	01	FB	00238	CALLS	#1, LIB\$FREE_EF		
	55	50	D0	0023F	MOVL	R0, STATUS		
	0C	55	E8	00242	BLBS	STATUS, 28\$		2413
00000000G	8F	55	D1	00245	CMPL	STATUS, #LIB\$_EF_ALRFRE		
		03	13	0024C	BEQL	28\$		
		00C7	31	0024E	BRW	38\$		
		67	A4	94 00251	CLRB	103(R4)		2415
		6C	A4	D5 00254	TSTL	108(R4)		2422
			17	13 00257	BEQL	31\$		
		6C	A4	9F 00259	PUSHAB	108(R4)		2426
04	AE	70	A4	3C 0025C	MOVZWL	112(R4), 4(SP)		2425



		04	AE	9F	00261	PUSHAB	4(SP)		
6B			02	FB	00264	CALLS	#2, LIB\$FREE_VM	2426	
55			50	D0	00267	MOVL	R0, STATUS		
		6C	A4	D4	0026A	CLRL	108(R4)	2427	
DE			55	E9	0026D	BLBC	STATUS, 27\$	2428	
50		00E4	C4	3C	00270	MOVZWL	228(R4), R0	2436	
			18	13	00275	BEQL	33\$		
		00B0	C4	9F	00277	PUSHAB	176(R4)	2439	
04	AE		50	D0	0027B	MOVL	R0, 4(SP)	2438	
		04	AE	9F	0027F	PUSHAB	4(SP)		
6B			02	FB	00282	CALLS	#2, LIB\$FREE_VM	2439	
55			50	D0	00285	MOVL	R0, STATUS		
		00E4	C4	B4	00288	CLRW	228(R4)	2440	
BF			55	E9	0028C	BLBC	STATUS, 27\$	2441	
53		00E8	C4	9E	0028F	MOVAB	232(R4), R3	2448	
			63	D5	00294	TSTL	(R3)		
			0F	13	00296	BEQL	34\$		
			63	DD	00298	PUSHL	(R3)	2450	
00000000G	00		01	FB	0029A	CALLS	#1, SYS\$CLOSE		
	55		50	D0	002A1	MOVL	R0, STATUS		
	71		55	E9	002A4	BLBC	STATUS, 38\$	2451	
	52		C4	9E	002A7	MOVAB	240(R4), R2	2458	
		00F0	62	D5	002AC	TSTL	(R2)		
			1C	13	002AE	BEQL	35\$		
			52	DD	002B0	PUSHL	R2	2460	
04	AE	5A	A4	3C	002B2	MOVZWL	90(R4), 4(SP)		
		04	AE	D6	002B7	INCL	4(SP)		
		04	AE	9F	002BA	PUSHAB	4(SP)		
00000000G	00		02	FB	002BD	CALLS	#2, LIB\$GET_VM		
	55		50	D0	002C4	MOVL	R0, STATUS		
			62	D4	002C7	CLRL	(R2)	2461	
	4C		55	E9	002C9	BLBC	STATUS, 38\$	2462	
			63	D5	002CC	TSTL	(R3)	2469	
			15	13	002CE	BEQL	36\$		
			53	DD	002D0	PUSHL	R3	2472	
04	AE	50	8F	9A	002D2	MOVZBL	#80, 4(SP)	2471	
		04	AE	9F	002D7	PUSHAB	4(SP)		
6B			02	FB	002DA	CALLS	#2, LIB\$FREE_VM	2472	
55			50	D0	002DD	MOVL	R0, STATUS		
			63	D4	002E0	CLRL	(R3)	2473	
	33		55	E9	002E2	BLBC	STATUS, 38\$	2474	
	52		C4	9E	002E5	MOVAB	236(R4), R2	2477	
		00EC	62	D5	002EA	TSTL	(R2)		
			15	13	002EC	BEQL	37\$		
			52	DD	002EE	PUSHL	R2	2480	
04	AE	44	8F	9A	002F0	MOVZBL	#68, 4(SP)	2479	
		04	AE	9F	002F5	PUSHAB	4(SP)		
6B			02	FB	002F8	CALLS	#2, LIB\$FREE_VM	2480	
55			50	D0	002FB	MOVL	R0, STATUS		
			62	D4	002FE	CLRL	(R2)	2481	
	15		55	E9	00300	BLBC	STATUS, 38\$	2482	
		04	AE	9F	00303	PUSHAB	PBCB	2489	
04	AE	014C	8F	3C	00306	MOVZWL	#332, 4(SP)		
		04	AE	9F	0030C	PUSHAB	4(SP)		
6B			02	FB	0030F	CALLS	#2, LIB\$FREE_VM		
55			50	D0	00312	MOVL	R0, STATUS		
04			55	E8	00315	BLBS	STATUS, 39\$		



SMG\$DISPLAY_LIN	SMG\$DISPLAY_LINKS - Virtual Display Linkages	K 16	16-Sep-1984 00:29:22	VAX-11 Bliss-32 V4.0-742	Page 66
1-096	SMG\$DELETE_PASTEBOARD - Delete Pasteboard	14-Sep-1984 13:09:43	[SMGRTL.SRC]SMGDISLIN.B32;1	(12)	

	50	55	D0 00318 38\$:	MOVL	STATUS, R0	:	2491
			04 0031B	RET		:	
	50	69	D0 0031C 39\$:	MOVL	(R9), R0	:	2498
00	44	AA	50 E5 0031F	BBCC	R0, PBD_V PB_AVAIL, 40\$	:	
		6A	D7 00324 40\$:	DECL	PBD_L_COUNT	:	2500
		04 AA40	D4 00326	CLRL	PBD_A-PBCB[R0]	:	2502
	50	01	D0 0032A	MOVL	#1, R0	:	2504
			04 0032D	RET		:	2506

; Routine Size: 814 bytes,      Routine Base: \_SMG\$CODE + 0A95

; 2257                    2507 1 !<BLF/PAGE>



```

: 2259      2508 1 %SBTTL 'SMG$CREATE_VIRTUAL_DISPLAY - Create Virtual Display'
: 2260      2509 1 GLOBAL ROUTINE SMG$CREATE_VIRTUAL_DISPLAY (
: 2261      2510 1      NUM_ROWS,      ! height
: 2262      2511 1      NUM_COLS,      ! width
: 2263      2512 1      NEW_DISPLAY_ID,
: 2264      2513 1      DISPLAY_ATTRIBUTES,
: 2265      2514 1      VIDEO_ATTRIBUTES,
: 2266      2515 1      CHAR_SET
: 2267      2516 1      ) =
: 2268      2517 1
: 2269      2518 1 ++
: 2270      2519 1 FUNCTIONAL DESCRIPTION:
: 2271      2520 1      This routine creates a new virtual display -- returning its
: 2272      2521 1      assigned display_id. Its initial contents are blanks with
: 2273      2522 1      video attributes set to those specified or zero. The cursor
: 2274      2523 1      will be at row 1 column 1.
: 2275      2524 1
: 2276      2525 1 CALLING SEQUENCE:
: 2277      2526 1
: 2278      2527 1      ret_status.wlc.v = SMG$CREATE_VIRTUAL_DISPLAY (
: 2279      2528 1      NUM_ROWS.rl.r,      ! Height
: 2280      2529 1      NUM_COLS.rl.r,      ! Width
: 2281      2530 1      NEW_DISPLAY_ID.wl.r
: 2282      2531 1      [,DISPLAY_ATTRIBUTES.rl.r]
: 2283      2532 1      [,VIDEO_ATTRIBUTES.rl.r]
: 2284      2533 1      [,CHAR_SET.rl.r])
: 2285      2534 1
: 2286      2535 1 FORMAL PARAMETERS:
: 2287      2536 1
: 2288      2537 1      NUM_ROWS.rl.r      Number of rows in new virtual display.
: 2289      2538 1
: 2290      2539 1      NUM_COLS.rl.r      Number of columns in new virtual display.
: 2291      2540 1
: 2292      2541 1      NEW_DISPLAY_ID.wl.r      Virtual display id of newly-created
: 2293      2542 1      virtual display.
: 2294      2543 1
: 2295      2544 1      DISPLAY_ATTRIBUTES.rl.r The default display attributes.
: 2296      2545 1
: 2297      2546 1      SMG$M_BORDER if virtual display is to be
: 2298      2547 1      displayed with a border.
: 2299      2548 1
: 2300      2549 1      SMG$M_TRUNC_ICON if an icon should be displayed
: 2301      2550 1      when text overflows the display bounds.
: 2302      2551 1
: 2303      2552 1      SMG$M_DISPLAY_CONTROLS if carriage controls (CR, LF,
: 2304      2553 1      TFF, VT, HT) should be displayed instead
: 2305      2554 1      of executed.
: 2306      2555 1      If omitted, none of the attributes will be set.
: 2307      2556 1
: 2308      2557 1      VIDEO_ATTRIBUTES.rl.r The default rendition code to be
: 2309      2558 1      applied to all output to this display unless
: 2310      2559 1      overridden on a particular output call.
: 2311      2560 1      If not supplied, default will be all zero (no
: 2312      2561 1      attributes).
: 2313      2562 1
: 2314      2563 1      Values:
: 2315      2564 1
  
```



```

: 2316      2565 1 | SMG$M_BLINK displays characters blinking.
: 2317      2566 1 |
: 2318      2567 1 | SMG$M_BOLD displays characters in
: 2319      2568 1 | higher-than-normal intensity.
: 2320      2569 1 |
: 2321      2570 1 | SMG$M_REVERSE displays characters in reverse
: 2322      2571 1 | video -- that is, using the
: 2323      2572 1 | opposite default rendition of
: 2324      2573 1 | the virtual display.
: 2325      2574 1 |
: 2326      2575 1 | SMG$M_UNDERLINE displays characters underlined.
: 2327      2576 1 |
: 2328      2577 1 | CHAR_SET.rb.r [Optional]. If provided, specifies the default
: 2329      2578 1 | character set to be used for this display.
: 2330      2579 1 | Recognized values are:
: 2331      2580 1 | SMG$C_UNITED_KINGDOM
: 2332      2581 1 | SMG$C_ASCII (default)
: 2333      2582 1 | SMG$C_SPEC_GRAPHICS
: 2334      2583 1 | SMG$C_ALT_CHAR
: 2335      2584 1 | SMG$C_ALT_GRAPHICS
: 2336      2585 1 |
: 2337      2586 1 | IMPLICIT INPUTS:
: 2338      2587 1 |
: 2339      2588 1 | NONE
: 2340      2589 1 |
: 2341      2590 1 | IMPLICIT OUTPUTS:
: 2342      2591 1 |
: 2343      2592 1 | NONE
: 2344      2593 1 |
: 2345      2594 1 | COMPLETION STATUS:
: 2346      2595 1 |
: 2347      2596 1 | $$$ NORMAL Normal successful completion
: 2348      2597 1 | LIB$_INSVIRMEM Insufficient virtual memory to allocate needed
: 2349      2598 1 | buffer.
: 2350      2599 1 | SMG$_INVARG Unrecognized Video Attributes
: 2351      2600 1 | or Unrecognized Display Attributes
: 2352      2601 1 | SMG$_WRONUMARG Wrong number of arguments.
: 2353      2602 1 |
: 2354      2603 1 | SIDE EFFECTS:
: 2355      2604 1 |
: 2356      2605 1 | NONE
: 2357      2606 1 | --
: 2358      2607 2 | BEGIN
: 2359      2608 2 | BUILTIN
: 2360      2609 2 | NULLPARAMETER;
: 2361      2610 2 |
: 2362      2611 2 | $SMG$VALIDATE_ARGCOUNT (3, 6); ! Test for right no. of args
: 2363      2612 2 |
: 2364      2613 3 | RETURN (SMG$$CREATE_VIRTUAL_DISPLAY(
: 2365      2614 3 | .NUM_ROWS,
: 2366      2615 3 | .NUM_COLS,
: 2367      2616 3 | .NEW_DISPLAY_ID, ! Gets the DCB address for the display created
: 2368      2617 4 | (IF NOT NULLPARAMETER(DISPLAY_ATTRIBUTES)
: 2369      2618 4 | THEN .DISPLAY_ATTRIBUTES
: 2370      2619 3 | ELSE UPLIT(0)),
: 2371      2620 4 | (IF NOT NULLPARAMETER(VIDEO_ATTRIBUTES)
: 2372      2621 4 | THEN .VIDEO_ATTRIBUTES
  
```



```

: 2373      2622  3      ELSE UPLIT(0) )
: 2374      2623  4      (IF NOT NULLPARAMETER(CHAR_SET)
: 2375      2624  4      THEN .CHAR_SET
: 2376      2625  2      ELSE UPLIT(0) ) ) );
: 2377      2626  2
: 2378      2627  1      END;
                                ! Routine SMG$CREATE_VIRTUAL_DISPLAY

```

```

                                00DC3      .BLKB      1
                                00000000 00DC4 P.AAC:  .LONG      0
                                00000000 00DC8 P.AAD:  .LONG      0
                                00000000 00DCC P.AAE:  .LONG      0

```

50	6C	03	83	00002	.ENTRY	SMG\$CREATE_VIRTUAL_DISPLAY, Save nothing	2509
	03	50	91	00006	SUBB3	#3, (AP), DIFF	2611
		08	1B	00009	CMPB	DIFF, #3	
	50	00000000G	8F	D0 0000B	BLEQU	1\$	
			04	00012	MOVL	#SMG\$_WRONUMARG, R0	
	06		6C	91 00013	RET		
			0A	1F 00016	CMPB	(AP), #6	2623
		18	AC	D5 00018	BLSSU	2\$	
			05	13 0001B	TSTL	24(AP)	
		18	AC	DD 0001D	BEQL	2\$	
			06	11 00020	PUSHL	CHAR_SET	2624
	50	D7	AF	9E 00022	BRB	3\$	
			50	DD 00026	MOVAB	P.AAE, R0	2625
	05		6C	91 00028	PUSHL	R0	
			0A	1F 0002B	CMPB	(AP), #5	2620
		14	AC	D5 0002D	BLSSU	4\$	
			05	13 00030	TSTL	20(AP)	
		14	AC	DD 00032	BEQL	4\$	
			06	11 00035	PUSHL	VIDEO_ATTRIBUTES	2621
	50	BE	AF	9E 00037	BRB	5\$	
			50	DD 0003B	MOVAB	P.AAD, R0	2622
	04		6C	91 0003D	PUSHL	R0	
			0A	1F 00040	CMPB	(AP), #4	2617
		10	AC	D5 00042	BLSSU	6\$	
			05	13 00045	TSTL	16(AP)	
		10	AC	DD 00047	BEQL	6\$	
			06	11 0004A	PUSHL	DISPLAY_ATTRIBUTES	2618
	50	A5	AF	9E 0004C	BRB	7\$	
			50	DD 00050	MOVAB	P.AAC, R0	2619
	7E	08	AC	7D 00052	PUSHL	R0	
			04	DD 00056	MOVQ	NUM_COLS, -(SP)	2615
	0000V	CF	06	FB 00059	PUSHL	NUM_ROWS	2614
			04	0005E	CALLS	#6, SMG\$CREATE_VIRTUAL_DISPLAY	
					RET		2627

```

; Routine Size: 95 bytes,   Routine Base: _SMG$CODE + 0DD0

```

```

; 2379      2628  1 !<BLF/PAGE>

```



```

2381 2629 1 XSBTTL 'SMG$DELETE_VIRTUAL_DISPLAY - Delete virtual display'
2382 2630 1 GLOBAL ROUTINE SMG$DELETE_VIRTUAL_DISPLAY ( DISPLAY_ID ) =
2383 2631 1 ++
2384 2632 1 FUNCTIONAL DESCRIPTION:
2385 2633 1
2386 2634 1 This routine deletes a virtual display. It is automatically
2387 2635 1 "unpasted" from any pasteboards on which it is pasted and
2388 2636 1 its associated buffer space is deallocated.
2389 2637 1
2390 2638 1 CALLING SEQUENCE:
2391 2639 1
2392 2640 1 ret_status.wlc.v = SMG$DELETE_VIRTUAL_DISPLAY (DISPLAY_ID.rl.r )
2393 2641 1
2394 2642 1 FORMAL PARAMETERS:
2395 2643 1
2396 2644 1 DISPLAY_ID.rl.r Id of virtual display to be deleted.
2397 2645 1
2398 2646 1 IMPLICIT INPUTS:
2399 2647 1
2400 2648 1 NONE
2401 2649 1
2402 2650 1 IMPLICIT OUTPUTS:
2403 2651 1
2404 2652 1 NONE
2405 2653 1
2406 2654 1 COMPLETION STATUS:
2407 2655 1
2408 2656 1 SS$ NORMAL Normal successful completion
2409 2657 1 SMG$_INVDIS_ID Invalid display id.
2410 2658 1 SMG$_WRONUMARG Wrong number of arguments.
2411 2659 1
2412 2660 1 SIDE EFFECTS:
2413 2661 1
2414 2662 1 NONE
2415 2663 1 --
2416 2664 2 BEGIN
2417 2665 2 LOCAL
2418 2666 2 STATUS, ! Status of subroutine calls
2419 2667 2 CURR_PP : REF $PP_DECL, ! Addr of current pasting packet
2420 2668 2 DCB : REF $DCB_DECL; ! Addr of display control block
2421 2669 2
2422 2670 2 $SMG$VALIDATE_ARGCOUNT (1, 1); ! Test for right no. of args
2423 2671 2
2424 2672 2 $SMG$GET_DCB ( .DISPLAY_ID, DCB); ! Get DCB address
2425 2673 2
2426 2674 2 CURR_PP = .DCB [DCB_A_PP_NEXT];
2427 2675 2
2428 2676 2 !+
2429 2677 2 Loop through all pasteboards we're pasted to, undoing our linkage to
2430 2678 2 each.
2431 2679 2 !-
2432 2680 2
2433 2681 2 WHILE .CURR_PP NEQ DCB [DCB_A_PP_NEXT] ! While any remain...
2434 2682 2 DO
2435 2683 2 BEGIN ! Overall loop
2436 2684 2 LOCAL
2437 2685 2 PBCB : REF $PBCB_DECL; ! Addr of pasteboard control blk

```



```

2438      2686      3
2439      2687      3      PBCB = .CURR_PP [PP_A_PBCB_ADDR];
2440      2688      3
2441      2689      3      !+
2442      2690      3      ! Update pasting packet pointer to next pasting packet, before
2443      2691      3      ! the unpaste operation makes current on go away.
2444      2692      3      !-
2445      2693      3      CURR_PP = .CURR_PP [PP_A_NEXT_DCB];
2446      2694      3
2447      2695      3      !+
2448      2696      3      ! Now we can unpaste this linkage.
2449      2697      3      !-
2450      2698      3      IF NOT (STATUS = SMG$$UNPASTE_VIRTUAL_DISPLAY (
2451      2699      3          .DCB,
2452      2700      3          .PBCB ))
2453      2701      3      THEN
2454      2702      3          RETURN (.STATUS);
2455      2703      3
2456      2704      3      END;      ! Overall loop
2457      2705      3
2458      2706      3      !+
2459      2707      3      ! Having successfully severed our linkage with all the pasteboards to
2460      2708      3      ! to which we were pasted, we can now get rid of the DCB itself.
2461      2709      3      ! Before we can delete this DCB we must check to see if there is a
2462      2710      3      ! backup DCB in existence. If so, call ourselves recursively to delete
2463      2711      3      ! the backup DCB first.
2464      2712      3      !-
2465      2713      3      IF .DCB [DCB_A_BACKUP_DCB] NEQ 0
2466      2714      3      THEN
2467      2715      3          IF NOT ( STATUS =SMG$DELETE_VIRTUAL_DISPLAY (
2468      2716      3              DCB [DCB_A_BACKUP_DCB]))
2469      2717      3      THEN
2470      2718      3          RETURN (.STATUS);
2471      2719      3
2472      2720      3      ! One remaining chore is to first release the buffer areas whose
2473      2721      3      ! addresses are in the DCB. Recall that the two buffer (text and
2474      2722      3      ! attr) were initially allocated as a double-size buffer and split in
2475      2723      3      ! two. This means we can return both at once by supplying the address
2476      2724      3      ! of the the text buffer and a length equal to twice its size.
2477      2725      3      !-
2478      2726      3      IF NOT (STATUS = LIB$FREE_VM ( %REF (2* .DCB [DCB_L_BUFSIZE]),
2479      2727      3          DCB [DCB_A_TEXT_BUF]))
2480      2728      3      THEN
2481      2729      3          RETURN (.STATUS);
2482      2730      3
2483      2731      3      !+
2484      2732      3      ! Free the line characteristics vector
2485      2733      3      !-
2486      2734      3      IF NOT (STATUS = LIB$FREE_VM (%REF ( .DCB [DCB_W_NO_ROWS] +1),
2487      2735      3          DCB [DCB_A_LINE_CHAR]))
2488      2736      3      THEN
2489      2737      3          RETURN ( .STATUS);
2490      2738      3
2491      2739      3      !+
2492      2740      3      ! Free the char_set buffer if there is one.
2493      2741      3      !-
2494      2742      3      IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
2495      2743      3      THEN

```



```

2495 2743 3      IF NOT (STATUS = LIB$FREE_VM ( DCB [DCB_L_BUFSIZE],
2496 2744 3      DCB [DCB_A_CHAR_SET_BUF]))
2497 2745 2      THEN
2498 2746 2      RETURN (.STATUS);
2499 2747 2
2500 2748 2      +
2501 2749 2      - If we have a dynamic string containing a border label, free the string
2502 2750 2
2503 2751 2      IF .DCB [DCB_V_BORDERED]
2504 2752 2      THEN
2505 2753 2      BEGIN ! Bordered
2506 2754 2      LOCAL
2507 2755 2      DESC : REF BLOCK [8,BYTE]; ! Pointer to dynamic string
2508 2756 2      ! descriptor in DCB
2509 2757 2      DESC = DCB [DCB_Q_LABEL_DESC];
2510 2758 2      IF .DESC [DESC$A_POINTER] NEQ 0
2511 2759 2      THEN
2512 2760 2      IF NOT (STATUS = LIB$FREE1_DD ( .DESC))
2513 2761 2      THEN
2514 2762 2      RETURN (.STATUS);
2515 2763 2      END; ! Bordered
2516 2764 2      +
2517 2765 2      Now the DCB itself...
2518 2766 2      Before freeing this area, we clobber the byte that makes it
2519 2767 2      recognizable as a DCB. That way, if someone inadvertantly tries to
2520 2768 2      pass us this DCB address as a DCB after having deleted the virtual
2521 2769 2      display, we can tell that it no longer is a valid DCB.
2522 2770 2      -
2523 2771 2      DCB [DCB_B_STRUCT_TYPE] = 0;
2524 2772 2      RETURN (LIB$FREE_VM (%REF (DCB_K_SIZE), DCB ));
2525 2773 2
2526 2774 1      END; ! Routine SMG$DELETE_VIRTUAL_DISPLAY
  
```

			001C 00000	.ENTRY	SMG\$DELETE_VIRTUAL_DISPLAY, Save R2,R3,R4	2630
	54	00000000G	00 9E 00002	MOVAB	LIB\$FREE_VM, R4	
	5E		08 C2 00009	SUBL2	#8, SP	
	01		6C 91 0000C	CMPB	(AP), #1	2670
			08 13 0000F	BEQL	1\$	
	50	00000000G	8F D0 00011	MOVL	#SMG\$_WRONUMARG, R0	
			04 00018	RET		
	50	04	BC D0 00019 1\$:	MOVL	@DISPLAY_ID, R0	2672
04	BC	38	A0 D1 0001D	CMPL	56(R0), @DISPLAY_ID	
			06 12 00022	BNEQ	2\$	
	11	44	A0 91 00024	CMPB	68(R0), #17	
			08 13 00028	BEQL	3\$	
	50	00000000G	8F D0 0002A 2\$:	MOVL	#SMG\$_INVDIS_ID, R0	
			04 00031	RET		
	04	AE	BC D0 00032 3\$:	MOVL	@DISPLAY_ID, DCB	
	52	04	AE D0 00037	MOVL	DCB, R2	2674
	53	20	A2 D0 0003B	MOVL	32(R2), CURR_PP	
	51	20	A2 9E 0003F 4\$:	MOVAB	32(R2), R1	2681
	51		53 D1 00043	CMPL	CURR_PP, R1	
			14 13 00046	BEQL	5\$	



	51	14	A3	D0	00048	MOVL	20(CURR_PP), PBCB	2687
	53		63	D0	0004C	MOVL	(CURR_PP), CURR_PP	2693
			51	DD	0004F	PUSHL	PBCB	2700
			52	DD	00051	PUSHL	R2	2699
0000V	CF		02	FB	00053	CALLS	#2, SMG\$UNPASTE_VIRTUAL_DISPLAY	
	E4		50	E8	00058	BLBS	STATUS, 4\$	2698
				04	0005B	RET		2702
		40	A2	D5	0005C	TSTL	64(R2)	2712
			0A	13	0005F	BEQL	6\$	
		40	A2	9F	00061	PUSHAB	64(R2)	2715
98	AF		01	FB	00064	CALLS	#1, SMG\$DELETE_VIRTUAL_DISPLAY	
	61		50	E9	00068	BLBC	STATUS, 9\$	
		10	A2	9F	0006B	PUSHAB	16(R2)	2726
04	AE	3C	01	78	0006E	ASHL	#1, 60(R2), 4(SP)	2725
		04	AE	9F	00074	PUSHAB	4(SP)	
	64		02	FB	00077	CALLS	#2, LIB\$FREE_VM	2726
	4F		50	E9	0007A	BLBC	STATUS, 9\$	
		4C	A2	9F	0007D	PUSHAB	76(R2)	2734
04	AE	02	A2	3C	00080	MOVZWL	2(R2), 4(SP)	2733
		04	AE	D6	00085	INCL	4(SP)	
		04	AE	9F	00088	PUSHAB	4(SP)	
	64		02	FB	0008B	CALLS	#2, LIB\$FREE_VM	2734
	3B		50	E9	0008E	BLBC	STATUS, 9\$	
		18	A2	D5	00091	TSTL	24(R2)	2741
			0C	13	00094	BEQL	7\$	
		18	A2	9F	00096	PUSHAB	24(R2)	2744
		3C	A2	9F	00099	PUSHAB	60(R2)	2743
	64		02	FB	0009C	CALLS	#2, LIB\$FREE_VM	2744
	2A		50	E9	0009F	BLBC	STATUS, 9\$	
	15	2F	A2	E9	000A2	BLBC	47(R2), 8\$	2751
	51	08	A2	9E	000A6	MOVAB	8(R2), DESC	2757
		04	A1	D5	000AA	TSTL	4(DESC)	2758
			0C	13	000AD	BEQL	8\$	
			51	DD	000AF	PUSHL	DESC	2760
00000000G	00		01	FB	000B1	CALLS	#1, LIB\$FREE1_DD	
	11		50	E9	000B8	BLBC	STATUS, 9\$	
		44	A2	94	000BB	CLRB	68(R2)	2771
		04	AE	9F	000BE	PUSHAB	DCB	2772
04	AE	70	8F	9A	000C1	MOVZBL	#112, 4(SP)	
		04	AE	9F	000C6	PUSHAB	4(SP)	
	64		02	FB	000C9	CALLS	#2, LIB\$FREE_VM	
			04	000CC	9\$:	RET		2774

; Routine Size: 205 bytes, Routine Base: \_SMG\$CODE + 0E2F

; 2527 2775 1 !<BLF/PAGE>



```

2529 2776 1 %SBTTL 'SMG$GET_DISPLAY_ATTR - Get display attributes'
2530 2777 1 GLOBAL ROUTINE SMG$GET_DISPLAY_ATTR (
2531 2778 1     DISPLAY_ID,
2532 2779 1     HEIGHT,
2533 2780 1     WIDTH,
2534 2781 1     DISPLAY_ATTRIBUTES,
2535 2782 1     VIDEO_ATTRIBUTES,
2536 2783 1     CHAR_SET
2537 2784 1 ) =
2538 2785 1
2539 2786 1 ++
2540 2787 1 FUNCTIONAL DESCRIPTION:
2541 2788 1     This routine returns attributes of the virtual display.
2542 2789 1
2543 2790 1 CALLING SEQUENCE:
2544 2791 1
2545 2792 1     ret_status.wlc.v = SMG$GET_DISPLAY_ATTR (
2546 2793 1         DISPLAY_ID.rl.r,
2547 2794 1         HEIGHT.wl.r,
2548 2795 1         WIDTH.wl.r,
2549 2796 1         [,DISPLAY_ATTRIBUTES.wl.r]
2550 2797 1         [,VIDEO_ATTRIBUTES.wl.r]
2551 2798 1         [,CHAR_SET.wl.r])
2552 2799 1
2553 2800 1 FORMAL PARAMETERS:
2554 2801 1
2555 2802 1     DISPLAY_ID.rl.r      The id of the display for which the
2556 2803 1                          information is requested.
2557 2804 1
2558 2805 1     HEIGHT.wl.r          Height of display in rows
2559 2806 1
2560 2807 1     WIDTH.wl.r           Width of display in columns
2561 2808 1
2562 2809 1     DISPLAY_ATTRIBUTES.wl.r Optional. If provided, the current
2563 2810 1                          default display attributes will be returned.
2564 2811 1                          These may be:
2565 2812 1
2566 2813 1                          SMG$M_BORDER if display is displayed with a
2567 2814 1                          border.
2568 2815 1
2569 2816 1     VIDEO_ATTRIBUTES.wl.r Optional. If provided, the current
2570 2817 1                          default video attributes are returned. These
2571 2818 1                          values may be:
2572 2819 1
2573 2820 1                          SMG$M_BLINK    displays characters blinking.
2574 2821 1
2575 2822 1                          SMG$M_BOLD    displays characters in
2576 2823 1                          higher-than-normal intensity.
2577 2824 1
2578 2825 1                          SMG$M_REVERSE displays characters in reverse
2579 2826 1                          video -- that is, using the
2580 2827 1                          opposite default rendition of
2581 2828 1                          the virtual display.
2582 2829 1
2583 2830 1                          SMG$M_UNDERLINE displays characters underlined.
2584 2831 1
2585 2832 1     CHAR_SET.wb.r      Optional. If provided, the current default
  
```



```

: 2586      2833 1 | character set code is returned.
: 2587      2834 1 | Possible values are:
: 2588      2835 1 | SMG$C_UNITED_KINGDOM
: 2589      2836 1 | SMG$C_ASCII (default)
: 2590      2837 1 | SMG$C_SPEC_GRAPHICS
: 2591      2838 1 | SMG$C_ALT_CHAR
: 2592      2839 1 | SMG$C_ALT_GRAPHICS
: 2593      2840 1 |
: 2594      2841 1 | IMPLICIT INPUTS:
: 2595      2842 1 |     NONE
: 2596      2843 1 |
: 2597      2844 1 | IMPLICIT OUTPUTS:
: 2598      2845 1 |     NONE
: 2599      2846 1 |
: 2600      2847 1 | COMPLETION STATUS:
: 2601      2848 1 |
: 2602      2849 1 |     SSS_NORMAL      Normal successful completion
: 2603      2850 1 |     SMG$_WRONUMARG  Wrong number of arguments
: 2604      2851 1 |
: 2605      2852 1 | SIDE EFFECTS:
: 2606      2853 1 |
: 2607      2854 1 |     NONE
: 2608      2855 1 |
: 2609      2856 1 | --
: 2610      2857 1 | BEGIN
: 2611      2858 2 | BUILTIN
: 2612      2859 2 | NULLPARAMETER;
: 2613      2860 2 |
: 2614      2861 2 | LOCAL
: 2615      2862 2 |     DCB : REF $DCB_DECL;          ! Addr of display control block
: 2616      2863 2 |
: 2617      2864 2 |     $SMG$VALIDATE_ARGCOUNT (3, 6);      ! Test for right no. of args
: 2618      2865 2 |
: 2619      2866 2 |     $SMG$GET_DCB ( .DISPLAY_ID, DCB);    ! Get DCB address
: 2620      2867 2 |
: 2621      2868 2 |     .HEIGHT = .DCB [DCB_W_NO_ROWS];
: 2622      2869 2 |     .WIDTH  = .DCB [DCB_W_NO_COLS];
: 2623      2870 2 |
: 2624      2871 2 |
: 2625      2872 2 |     IF NOT NULLPARAMETER (DISPLAY_ATTRIBUTES)
: 2626      2873 2 |     THEN .DISPLAY_ATTRIBUTES = .DCB [DCB_B_DEF_DISPLAY_ATTR];
: 2627      2874 2 |
: 2628      2875 2 |     IF NOT NULLPARAMETER (VIDEO_ATTRIBUTES)
: 2629      2876 2 |     THEN .VIDEO_ATTRIBUTES = .DCB [DCB_B_DEF_VIDEO_ATTR];
: 2630      2877 2 |
: 2631      2878 2 |     IF NOT NULLPARAMETER (CHAR_SET)
: 2632      2879 2 |     THEN .CHAR_SET = .DCB [DCB_B_DEF_CHAR_SET];
: 2633      2880 2 |
: 2634      2881 2 |     RETURN (SS$_NORMAL);
: 2635      2882 1 | END;                                ! Routine SMG$GET_DISPLAY_ATTR

```



SMG\$DISPLAY_LIN	SMG\$DISPLAY_LINKS - Virtual Display Linkages	16-Sep-1984 00:29:22	VAX-11 Bliss-32 V4.0-742	Page 76
1-096	SMG\$GET_DISPLAY_ATTR - Get display attributes	14-Sep-1984 13:09:43	[SMGRTL.SRC]SMGDISLIN.B32;1	(15)

03	50	91	00006	CMPB	DIFF, #3	:	
	08	1B	00009	BLEQU	1\$	:	
50	00000000G	8F	D0 0000B	MOVL	#SMG\$_WRONUMARG, R0	:	
		04	00012	RET		:	
04	50	04	BC D0 00013	1\$:	MOVL	@DISPLAY_ID, R0	2867
	BC	38	A0 D1 00017		CMPL	56(R0), @DISPLAY_ID	:
		06	12 0001C		BNEQ	2\$	:
	11	44	A0 91 0001E		CMPB	68(R0), #17	:
		08	13 00022		BEQL	3\$	:
50	00000000G	8F	D0 00024	2\$:	MOVL	#SMG\$_INVDIS_ID, R0	:
		04	0002B		RET		:
08	50	04	BC D0 0002C	3\$:	MOVL	@DISPLAY_ID, DCB	2869
	BC	02	A0 3C 00030		MOVZWL	2(DCB), @HEIGHT	:
0C	BC	06	A0 3C 00035		MOVZWL	6(DCB), @WIDTH	2870
	04		6C 91 0003A		CMPB	(AP), #4	2872
			0A 1F 0003D		BLSSU	4\$	:
		10	AC D5 0003F		TSTL	16(AP)	:
		05	13 00042		BEQL	4\$	:
10	BC	2F	A0 9A 00044		MOVZBL	47(DCB), @DISPLAY_ATTRIBUTES	2873
	05		6C 91 00049	4\$:	CMPB	(AP), #5	2875
			0A 1F 0004C		BLSSU	5\$	:
		14	AC D5 0004E		TSTL	20(AP)	:
		05	13 00051		BEQL	5\$	:
14	BC	2E	A0 9A 00053		MOVZBL	46(DCB), @VIDEO_ATTRIBUTES	2876
	06		6C 91 00058	5\$:	CMPB	(AP), #6	2878
			0A 1F 0005B		BLSSU	6\$	:
		18	AC D5 0005D		TSTL	24(AP)	:
		05	13 00060		BEQL	6\$	:
18	BC	30	A0 9A 00062		MOVZBL	48(DCB), @CHAR_SET	2879
	50		01 D0 00067	6\$:	MOVL	#1, R0	2881
		04	0006A		RET		2882

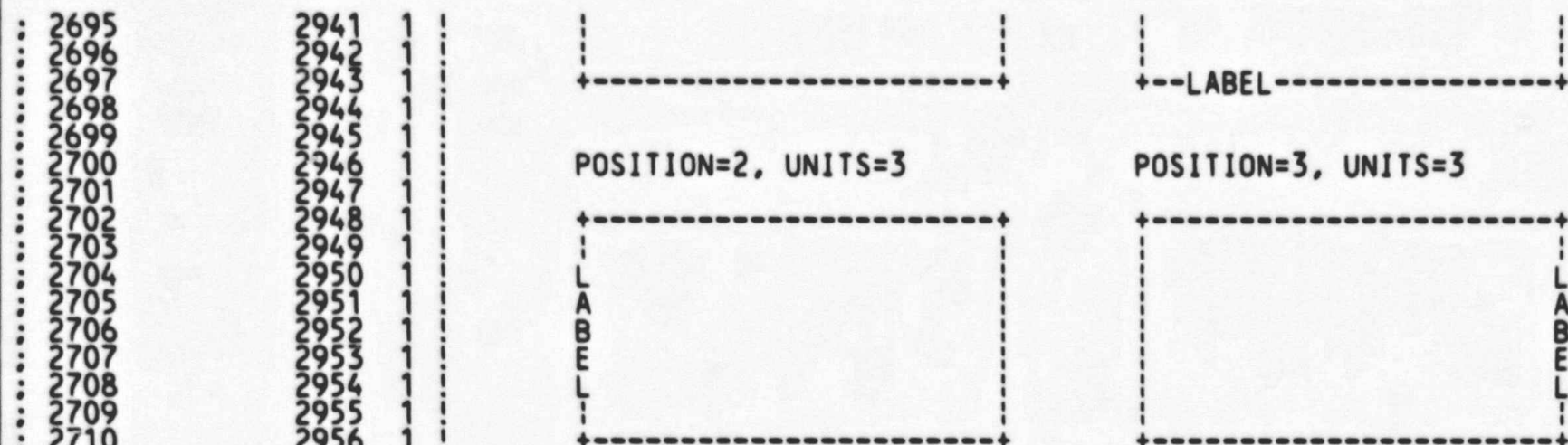
; Routine Size: 107 bytes, Routine Base: \_SMG\$CODE + 0EFC

; 2636 2883 1 !<BLF/PAGE>









# CALLING SEQUENCE:

```
ret_status.wlc.v = SMG$LABEL_BORDER (
    DISPLAY_ID.rl.r
    [,LABEL_TEXT.rt.dx]
    [,POSITION.rl.r]
    [,UNITS.rl.r]
    [,RENDITION_SET.rl.r]
    [,RENDITION_COMPLEMENT.rl.r.]
    [,CHAR_SET.rl.r])
```

# FORMAL PARAMETERS:

DISPLAY\_ID.rl.r The display id of the virtual display whose border is to be labeled. This display must have the display attribute of SMG\$M\_BORDER.

LABEL\_TEXT.rt.dx [Optional]. If supplied becomes the new label for this display's border. If omitted, display becomes unlabeled.

POSITION.rl.r [Optional]. Specifies which border will contain label. If omitted, default to top border.

UNITS.rl.r [Optional]. Specifies where within the border the text label will start. If omitted, center in line indicated by POSITION

RENDITION\_SET.rl.r [Optional]. Each 1 bit attribute in this parameter causes the corresponding attribute to be set in the display. (See below for list of settable attributes.)

RENDITION\_COMPLEMENT.rl.r [Optional]. Each 1 bit attribute in this parameter causes the corresponding attribute to



2752 2998 1  
 2753 2999 1  
 2754 3000 1  
 2755 3001 1  
 2756 3002 1  
 2757 3003 1  
 2758 3004 1  
 2759 3005 1  
 2760 3006 1  
 2761 3007 1  
 2762 3008 1  
 2763 3009 1  
 2764 3010 1  
 2765 3011 1  
 2766 3012 1  
 2767 3013 1  
 2768 3014 1  
 2769 3015 1  
 2770 3016 1  
 2771 3017 1  
 2772 3018 1  
 2773 3019 1  
 2774 3020 1  
 2775 3021 1  
 2776 3022 1  
 2777 3023 1  
 2778 3024 1  
 2779 3025 1  
 2780 3026 1  
 2781 3027 1  
 2782 3028 1  
 2783 3029 1  
 2784 3030 1  
 2785 3031 1  
 2786 3032 1  
 2787 3033 1  
 2788 3034 1  
 2789 3035 1  
 2790 3036 1  
 2791 3037 1  
 2792 3038 1  
 2793 3039 1  
 2794 3040 1  
 2795 3041 1  
 2796 3042 1  
 2797 3043 1  
 2798 3044 1  
 2799 3045 1  
 2800 3046 1  
 2801 3047 1  
 2802 3048 1  
 2803 3049 1  
 2804 3050 1  
 2805 3051 1  
 2806 3052 1  
 2807 3053 1  
 2808 3054 1

be complemented in the display. (See  
 below for list of complementable  
 attributes.)

If the same bit is specified in both the RENDITION\_SET parameter  
 and in the RENDITION\_COMPLEMENT parameter, the application is  
 RENDITION\_SET followed by RENDITION complement. Using these two  
 parameters together the caller can exercise arbitrary and  
 independent control over each attribute on a single call. On an  
 attribute by attribute basis he can cause the following  
 transformations:

SET	COMPLEMENT	Action
---	---	
0	0	Attribute unchanged.
1	0	Attribute set to 'on'.
0	1	Attribute set to complement of current setting.
1	1	Attribute set to 'off'.

Attributes which can be manipulated in this manner are:

SMG\$M\_BLINK displays characters blinking.  
 SMG\$M\_BOLD displays characters in higher-than-normal  
 intensity.  
 SMG\$M\_REVERSE displays characters in reverse video -- that is,  
 using the opposite default rendition of the  
 virtual display.  
 SMG\$M\_UNDERLINE displays characters underlined.  
 CHAR\_SET.rl.r [Optional]. If provided, the character set to  
 be used in displaying the label.  
 Recognized values are:  
 SMG\$C\_UNITED\_KINGDOM  
 SMG\$C\_ASCII (default)  
 SMG\$C\_SPEC\_GRAPHICS  
 SMG\$C\_ALT\_CHAR  
 SMG\$C\_ALT\_GRAPHICS

#### IMPLICIT INPUTS:

None

#### IMPLICIT OUTPUTS:

None

#### COMPLETION STATUS:

SS\$ NORMAL Normal successful completion  
 SMG\$\_INVDIS\_ID Invalid virtual display id.  
 SMG\$\_INVARG Positioning and/or units when considered with  
 length of text results in a position that is  
 outside of the border area.  
 SMG\$\_WRONUMARG Wrong number of arguments.



```

2809 3055 1 | SIDE EFFECTS:
2810 3056 1 |
2811 3057 1 |
2812 3058 1 |     NONE
2813 3059 1 | --
2814 3060 2 | BEGIN
2815 3061 2 |
2816 3062 2 | LITERAL
2817 3063 2 |     K_SET_ARG = 5,
2818 3064 2 |     K_COMP_ARG = 6;
2819 3065 2 |
2820 3066 2 | BUILTIN
2821 3067 2 |     NULLPARAMETER;
2822 3068 2 |
2823 3069 2 | LOCAL
2824 3070 2 |     LUNITS,                ! Implicit or explicit UNITS
2825 3071 2 |     LPOS,                  ! Implicit or explicit POSITION
2826 3072 2 |     REND_CODE,             ! Rendition to be applied to
2827 3073 2 |                             ! border label
2828 3074 2 |     STATUS,                ! Status of subroutine calls
2829 3075 2 |     DESC : REF BLOCK [,BYTE], ! Pointer to dynamic string
2830 3076 2 |                             ! descriptor in DCB for border
2831 3077 2 |                             ! label.
2832 3078 2 |     DCB : REF $DCB_DECL;    ! Addr. of display control block
2833 3079 2 |
2834 3080 2 | $SMG$VALIDATE_ARGCOUNT (1, 7); ! Test for right no. of args
2835 3081 2 |
2836 3082 2 | $SMG$GET_DCB ( .DISPLAY_ID, DCB); ! Get addr of DCB
2837 3083 2 |
2838 3084 2 | +
2839 3085 2 | Get a copy of the label.
2840 3086 2 | -
2841 3087 2 |     DESC = DCB [DCB_Q_LABEL_DESC];
2842 3088 2 |
2843 3089 2 | IF NULLPARAMETER (LABEL_TEXT)
2844 3090 2 | THEN
2845 3091 2 |     BEGIN ! No text specified
2846 3092 2 |         RETURN (LIB$SFREE1_DD ( .DESC));
2847 3093 2 |     END; ! No text specified
2848 3094 2 |
2849 3095 2 | IF NOT (STATUS = LIB$SCOPY_DXDX (.LABEL_TEXT, .DESC))
2850 3096 2 | THEN
2851 3097 2 |     RETURN (.STATUS);
2852 3098 2 |
2853 3099 2 | +
2854 3100 2 | Check to see if combination of POSITION and UNITS fit.
2855 3101 2 | -
2856 3102 2 |     LPOS = ( IF NOT NULLPARAMETER (POSITION) THEN ..POSITION
2857 3103 2 |                 ELSE 0); ! Default to top row
2858 3104 2 |
2859 3105 2 | CASE .LPOS FROM SMG$K_TOP TO SMG$K_RIGHT OF
2860 3106 2 | SET
2861 3107 2 |     [SMG$K_TOP,SMG$K_BOTTOM]: ! Top or bottom row
2862 3108 2 |     BEGIN
2863 3109 2 |         LUNITS = ( IF NOT NULLPARAMETER (UNITS)
2864 3110 2 |                     THEN ..UNITS
2865 3111 2 |                     ELSE ! Center horizontally

```



```
2866      3112 6      ((.DCB[DCB_W_NO_COLS] -.DESC [DSC$W_LENGTH])
2867      3113      / 2) + 2);
2868      3114
2869      3115      IF .LUNITS LEQ 0      OR
2870      3116      .LUNITS + .DESC[DSC$W_LENGTH] GTR .DCB [DCB_W_NO_COLS] +2
2871      3117      THEN
2872      3118      BEGIN
2873      3119      LIB$FREE1 DD (.DESC) ; ! Release our dynamic string
2874      3120      RETURN (SMG$_INVARG);
2875      3121      END;
2876      3122      END;
2877      3123
2878      3124 [SMG$K_LEFT,SMG$K_RIGHT]:      ! Left or right column
2879      3125      BEGIN
2880      3126      LUNITS = ( IF NOT NULLPARAMETER (UNITS)
2881      3127      THEN ..UNITS
2882      3128      ELSE      ! Center vertically
2883      3129      ((.DCB[DCB_W_NO_ROWS] -.DESC[DSC$W_LENGTH])
2884      3130      / 2) + 2);
2885      3131
2886      3132      IF .LUNITS LEQ 0      OR
2887      3133      .LUNITS + .DESC[DSC$W_LENGTH] GTR .DCB [DCB_W_NO_ROWS] +2
2888      3134      THEN
2889      3135      BEGIN
2890      3136      LIB$FREE1 DD (.DESC) ; ! Release our dynamic string
2891      3137      RETURN (SMG$_INVARG);
2892      3138      END;
2893      3139      END;
2894      3140
2895      3141 [OUTRANGE]:
2896      3142      RETURN (SMG$_INVARG);
2897      3143      TES;
2898      3144
2899      3145      DCB [DCB_B_LABEL_POS] = .LPOS;
2900      3146      DCB [DCB_W_LABEL_UNITS] = .LUNITS;
2901      3147
2902      3148      !+
2903      3149      ! If UNITS parameter was omitted we centered the label. Make a note of
2904      3150      ! this fact so that if he later does a CHANGE_VIRTUAL_DISPLAY we can
2905      3151      ! again center it in its new "center".
2906      3152      DCB [DCB_V_LABEL_CENTER] = 0;
2907      3153      IF NULLPARAMETER (UNITS)
2908      3154      THEN
2909      3155      DCB [DCB_V_LABEL_CENTER] = 1;
2910      3156
2911      3157      !+
2912      3158      ! Calc. REND CODE as a function of callers rendition arguments and
2913      3159      ! the default rendition in the DCB.
2914      3160      !-
2915      3161      $SMG$SET_REND_CODE (K_SET_ARG, K_COMP_ARG);
2916      3162      ! macro to use caller's args if present
2917      3163
2918      3164      DCB [DCB_B_LABEL_REND] = .REND_CODE;
2919      3165
2920      3166      !+
2921      3167      ! Deal with alternate character set.
2922      3168      !-
```



```

2923 3169 2 IF NOT NULLPARAMETER(CHAR_SET)
2924 3170 2 THEN
2925 3171 2 BEGIN
2926 3172 2 CASE ..CHAR_SET FROM SMG$C_UNITED_KINGDOM
2927 3173 2 TO SMG$C_ALT_GRAPHICS OF
2928 3174 2 SET
2929 3175 2 [SMG$C_UNITED_KINGDOM,
2930 3176 2 SMG$C_ASCII,
2931 3177 2 SMG$C_SPEC_GRAPHICS,
2932 3178 2 SMG$C_ALT_CHAR,
2933 3179 2 SMG$C_ALT_GRAPHICS];
2934 3180 2 DCB [DCB_B_LABEL_CHAR_SET] = ..CHAR_SET;
2935 3181 2
2936 3182 2 [INRANGE, OUTRANGE]:
2937 3183 2 RETURN (SMG$_INVARG);
2938 3184 2
2939 3185 2 TES;
2940 3186 2 END
2941 3187 2 ELSE ! Use default for virtual display
2942 3188 2 DCB [DCB_B_LABEL_CHAR_SET] = .DCB [DCB_B_DEF_CHAR_SET];
2943 3189 2
2944 3190 2 DCB [DCB_V_BORDERED] = 1; ! Force bordered attribute in case it
2945 3191 2 ! wasn't previously.
2946 3192 2
2947 3193 2 We now need to recalculate the constants in the pasting packet.
2948 3194 2 We may be making the transition from unbordered to bordered, so
2949 3195 2 this virtual display now has a bigger footprint in the pasteboard
2950 3196 2 buffer, and some display which previously was not occluded may now be.
2951 3197 2 Even if we were previously bordered, the size and position of our
2952 3198 2 label may have changed.
2953 3199 2 If we are not batched at the display level, recalc. pasting packet
2954 3200 2 constants and initiate output. Else, just remember that we need to do
2955 3201 2 it later when batch level drops to zero.
2956 3202 2
2957 3203 2 IF .DCB [DCB_L_BATCH_LEVEL] EQL 0
2958 3204 2 THEN
2959 3205 2 BEGIN ! Do it now
2960 3206 2 IF NOT (STATUS = SMG$$RECALC_PP_FIELDS ( .DCB))
2961 3207 2 THEN
2962 3208 2 RETURN (.STATUS);
2963 3209 2
2964 3210 2 RETURN ( SMG$$CHECK_FOR_OUTPUT_DCB ( .DCB, SMG$C_LABEL_BORDER));
2965 3211 2 END ! Do it now
2966 3212 2 ELSE
2967 3213 2
2968 3214 2 BEGIN ! Defer the action
2969 3215 2 DCB [DCB_V_PP_MISMATCH] =1; ! Remember for later
2970 3216 2 END; ! Defer the action
2971 3217 2
2972 3218 2 RETURN (SS$_NORMAL);
2973 3219 1 END; ! Routine SMG$LABEL_BORDER

```



50	57	00000000G	00	9E	00002	MOVAB	LIB\$SFREE1 DD, R7	3080
	6C		01	83	00009	SUBB3	#1, (AP), DIFF	
	06		50	91	0000D	CMPB	DIFF, #6	
			08	1B	00010	BLEQU	1\$	
	50	00000000G	8F	D0	00012	MOVL	#SMG\$_WRONUMARG, R0	
			04	04	00019	RET		
04	50	04	BC	D0	0001A	1\$: MOVL	@DISPLAY ID, R0	3082
	BC	38	A0	D1	0001E	CMPL	56(R0), @DISPLAY_ID	
			06	12	00023	BNEQ	2\$	
	11	44	A0	91	00025	CMPB	68(R0), #17	
			08	13	00029	BEQL	3\$	
	50	00000000G	8F	D0	0002B	2\$: MOVL	#SMG\$_INVDIS_ID, R0	
			04	04	00032	RET		
	52	04	BC	D0	00033	3\$: MOVL	@DISPLAY ID, DCB	
	54	08	A2	9E	00037	MOVAB	8(R2), DESC	3087
	02		6C	91	0003B	CMPB	(AP), #2	3089
			05	1F	0003E	BLSSU	4\$	
		08	AC	D5	00040	TSTL	8(AP)	
			06	12	00043	BNEQ	5\$	
	67		54	DD	00045	4\$: PUSHL	DESC	3092
			01	FB	00047	CALLS	#1, LIB\$SFREE1_DD	
			04	04	0004A	RET		
			54	DD	0004B	5\$: PUSHL	DESC	3095
		08	AC	DD	0004D	PUSHL	LABEL TEXT	
00000000G	00		02	FB	00050	CALLS	#2, LIB\$SCOPY_DXDX	
	56		50	D0	00057	MOVL	R0, STATUS	
	03		56	E8	0005A	BLBS	STATUS, 6\$	
		010B	31	0005D	BRW	29\$		
	03		6C	91	00060	6\$: CMPB	(AP), #3	3102
			0B	1F	00063	BLSSU	7\$	
		0C	AC	D5	00065	TSTL	12(AP)	
			06	13	00068	BEQL	7\$	
	55	0C	BC	D0	0006A	MOVL	@POSITION, LPOS	
			02	11	0006E	BRB	8\$	
			55	D4	00070	7\$: CLRL	LPOS	
			55	CF	00072	8\$: CASEL	LPOS, #0, #3	3105
0039	0039	000A	000A	00076	9\$: .WORD		10\$-9\$,- 10\$-9\$,- 13\$-9\$,- 13\$-9\$	
			68	11	0007E	BRB	18\$	3142
	04		6C	91	00080	10\$: CMPB	(AP), #4	3109
			0B	1F	00083	BLSSU	11\$	
		10	AC	D5	00085	TSTL	16(AP)	
			06	13	00088	BEQL	11\$	
	53	10	BC	D0	0008A	MOVL	@UNITS, LUNITS	3110
			11	11	0008E	BRB	12\$	
	50	06	A2	3C	00090	11\$: MOVZWL	6(DCB), R0	3112
	51		64	3C	00094	MOVZWL	(DESC), R1	
	50		51	C2	00097	SUBL2	R1, R0	
	50		02	C6	0009A	DIVL2	#2, R0	3113
	53	02	A0	9E	0009D	MOVAB	2(R0), LUNITS	
			40	15	000A1	12\$: BLEQ	17\$	3115
	51		64	3C	000A3	MOVZWL	(DESC), R1	3116
	51		53	C0	000A6	ADDL2	LUNITS, R1	
	50	06	A2	3C	000A9	MOVZWL	6(DCB), R0	
			2C	11	000AD	BRB	16\$	



04			6C	91	000AF	13\$:	CMPB	(AP), #4		3126
			0B	1F	000B2		BLSSU	14\$		
		10	AC	D5	000B4		TSTL	16(AP)		
			06	13	000B7		BEQL	14\$		
53		10	BC	D0	000B9		MOVL	@UNITS, LUNITS		3127
			10	11	000BD		BRB	15\$		
53		02	A2	3C	000BF	14\$:	MOVZWL	2(DCB), R3		3129
50			64	3C	000C3		MOVZWL	(DESC), R0		
53			50	C2	000C6		SUBL2	R0, R3		
53			02	C6	000C9		DIVL2	#2, R3		3130
53			02	C0	000CC		ADDL2	#2, LUNITS		
			12	15	000CF	15\$:	BLEQ	17\$		3132
51			64	3C	000D1		MOVZWL	(DESC), R1		3133
51			53	C0	000D4		ADDL2	LUNITS, R1		
50		02	A2	3C	000D7		MOVZWL	2(DCB), R0		
50			02	C0	000DB	16\$:	ADDL2	#2, R0		
50			51	D1	000DE		CMPL	R1, R0		
			07	15	000E1		BLEQ	19\$		
			54	DD	000E3	17\$:	PUSHL	DESC		3136
67			01	FB	000E5		CALLS	#1, LIB\$SFREE1_DD		
			57	11	000E8	18\$:	BRB	25\$		3137
			55	90	000EA	19\$:	MOVB	LPOS, 49(DCB)		3145
31	A2		53	B0	000EE		MOVW	LUNITS, 44(DCB)		3146
2C	A2		04	8A	000F2		BICB2	#4, 52(DCB)		3152
34	A2		6C	91	000F6		CMPB	(AP), #4		3153
	04		05	1F	000F9		BLSSU	20\$		
		10	AC	D5	000FB		TSTL	16(AP)		
			04	12	000FE		BNEQ	21\$		
34	A2		04	88	00100	20\$:	BISB2	#4, 52(DCB)		3155
	50		A2	9A	00104	21\$:	MOVZBL	46(DCB), REND_CODE		3161
	05		6C	91	00108		CMPB	(AP), #5		
			09	1F	0010B		BLSSU	22\$		
		14	AC	D5	0010D		TSTL	20(AP)		
			04	13	00110		BEQL	22\$		
	50		BC	C8	00112		BISL2	@RENDITION_SET, REND_CODE		
	06		6C	91	00116	22\$:	CMPB	(AP), #6		
			09	1F	00119		BLSSU	23\$		
		18	AC	D5	0011B		TSTL	24(AP)		
			04	13	0011E		BEQL	23\$		
	50		BC	CC	00120		XORL2	@RENDITION_COMPLEMENT, REND_CODE		
33	A2		50	90	00124	23\$:	MOVB	REND_CODE, 51(DCB)		3164
	07		6C	91	00128		CMPB	(AP), #7		3169
			23	1F	0012B		BLSSU	27\$		
		1C	AC	D5	0012D		TSTL	28(AP)		
			1E	13	00130		BEQL	27\$		
		1C	BC	CF	00132		CASEL	@CHAR_SET, #0, #4		3172
00										
0012			0012		00137	24\$:	.WORD	26\$-24\$,-		
			0012		0013F			26\$-24\$,-		
								26\$-24\$,-		
								26\$-24\$,-		
								26\$-24\$,-		
								26\$-24\$,-		
	50	00000000G	8F	D0	00141	25\$:	MOVL	#SMG\$_INVARG, R0		3183
				04	00148		RET			
32	A2		1C	BC	90	00149	26\$:	MOVB	@CHAR_SET, 50(DCB)	3180
				05	11	0014E		BRB	28\$	3169
32	A2		30	A2	90	00150	27\$:	MOVB	48(DCB), 50(DCB)	3187
2F	A2		01	88	00155	28\$:	BISB2	#1, 47(DCB)		3189



SMG\$DISPLAY\_LIN SMG\$DISPLAY LINKS - Virtual Display Linkages  
1-096 SMG\$LABEL\_BORDER - Specify label for border

E 2  
16-Sep-1984 00:29:22  
14-Sep-1984 13:09:43

VAX-11 Bliss-32 V4.0-742  
[SMGRTL.SRC]SMGDISLIN.B32;1

Page 85  
(16)

		1C	A2	D5	00159	TSTL	28(DCB)	:	3202
			1D	12	0015C	BNEQ	31\$	:	
			52	DD	0015E	PUSHL	DCB	:	3205
0000V	CF		01	FB	00160	CALLS	#1, SMG\$\$RECALC_PP_FIELDS	:	
	56		50	D0	00165	MOVL	R0, STATUS	:	
	04		56	E8	00168	BLBS	STATUS, 30\$	:	
	50		56	D0	0016B	MOVL	STATUS, R0	:	3207
				04	0016E	RET		:	
			1C	DD	0016F	PUSHL	#28	:	3209
			52	DD	00171	PUSHL	DCB	:	
00000000G	00		02	FB	00173	CALLS	#2, SMG\$\$CHECK_FOR_OUTPUT_DCB	:	
				04	0017A	RET		:	
34	A2		08	88	0017B	BISB2	#8, 52(DCB)	:	3215
	50		01	D0	0017F	MOVL	#1, R0	:	3218
				04	00182	RET		:	3219

; Routine Size: 387 bytes, Routine Base: \_SMG\$CODE + 0F67

; 2974 3220 1 !<BLF/PAGE>



```

: 2976 3221 1 %SBTTL 'SMG$MOVE_VIRTUAL_DISPLAY - Move previously pasted virtual display'
: 2977 3222 1 GLOBAL ROUTINE SMG$MOVE_VIRTUAL_DISPLAY (
: 2978 3223 1     DISPLAY_ID,
: 2979 3224 1     PASTEBOARD_ID,
: 2980 3225 1     PASTEBOARD_ROW,
: 2981 3226 1     PASTEBOARD_COL
: 2982 3227 1 ) =
: 2983 3228 1
: 2984 3229 1 ++
: 2985 3230 1 FUNCTIONAL DESCRIPTION:
: 2986 3231 1     The specified virtual display is moved with respect to the
: 2987 3232 1     position where it is currently pasted to the specified pasteboard
: 2988 3233 1     preserving the pasting order. If the display is not currently
: 2989 3234 1     pasted, it is pasted at the top of the pasting order in the
: 2990 3235 1     position specified.
: 2991 3236 1     This call is not permitted while display batching is in effect.
: 2992 3237 1
: 2993 3238 1 CALLING SEQUENCE:
: 2994 3239 1     ret_status.wlc.v = SMG$MOVE_VIRTUAL_DISPLAY (
: 2995 3240 1         DISPLAY_ID.rl.r,
: 2996 3241 1         PASTEBOARD_ID.rl.r,
: 2997 3242 1         PASTEBOARD_ROW.rl.r,
: 2998 3243 1         PASTEBOARD_COL.rl.r)
: 2999 3244 1
: 3000 3245 1 FORMAL PARAMETERS:
: 3001 3246 1
: 3002 3247 1     DISPLAY_ID.rl.r     Id of virtual display to be moved.
: 3003 3248 1
: 3004 3249 1     PASTEBOARD_ID.rl.r  The pasteboard id of the pasteboard on
: 3005 3250 1                        which the movement is to take place.
: 3006 3251 1
: 3007 3252 1     PASTEBOARD_ROW.rl.r Row on pasteboard which is to contain
: 3008 3253 1                        row 1 of the specified virtual display.
: 3009 3254 1
: 3010 3255 1     PASTEBOARD_COL.rl.r Column on pasteboard which is to contain
: 3011 3256 1                        column 1 of the specified virtual
: 3012 3257 1                        display.
: 3013 3258 1
: 3014 3259 1 IMPLICIT INPUTS:
: 3015 3260 1
: 3016 3261 1     None
: 3017 3262 1
: 3018 3263 1 IMPLICIT OUTPUTS:
: 3019 3264 1
: 3020 3265 1     None
: 3021 3266 1
: 3022 3267 1 COMPLETION STATUS:
: 3023 3268 1
: 3024 3269 1     SSS NORMAL      Normal successful completion
: 3025 3270 1     SMG$-INVDIS_ID  Invalid virtual display id.
: 3026 3271 1     SMG$-INVPAS_ID  Invalid pasteboard id.
: 3027 3272 1     SMG$-WRONUMARG  Wrong number of arguments.
: 3028 3273 1     SMG$-ILLBATFNC  Display is being batched, this operation is illegal.
: 3029 3274 1
: 3030 3275 1 SIDE EFFECTS:
: 3031 3276 1
: 3032 3277 1
  
```



```

3033 3278 1 | NONE
3034 3279 1 | --
3035 3280 2 | BEGIN
3036 3281 2 | BUILTIN
3037 3282 2 |   AP,
3038 3283 2 |   CALLG;
3039 3284 2 |
3040 3285 2 | LOCAL
3041 3286 2 |   STATUS,                ! Status of subroutine calls
3042 3287 2 |
3043 3288 2 |   PP      : REF $PP_DECL,  ! Addr of the pasting packet
3044 3289 2 |   DCB     : REF $DCB_DECL, ! Addr. of display control block
3045 3290 2 |   PBCB    : REF $PBCB_DECL; ! Addr of pasteboard control block
3046 3291 2 |
3047 3292 2 | $SMG$VALIDATE_ARGCOUNT (4, 4); ! Test for right no. of args
3048 3293 2 |
3049 3294 2 | +
3050 3295 2 | Get addresses of associated virtual display control block and
3051 3296 2 | pasteboard control block, validating both the display id and the
3052 3297 2 | pasteboard id.
3053 3298 2 | -
3054 3299 2 |   $SMG$GET_DCB ( .DISPLAY_ID, DCB);      ! Get addr of DCB
3055 3300 2 |   $SMG$GET_PBCB ( .PASTEBOARD_ID, PBCB); ! Get addr of PBCB
3056 3301 2 |
3057 3302 2 | +
3058 3303 2 | Give an error if the display is batched.
3059 3304 2 | -
3060 3305 2 |
3061 3306 2 | IF .DCB[DCB_L_BATCH_LEVEL] NEQ 0
3062 3307 2 | THEN
3063 3308 2 |   RETURN SMG$_ILLBATFNC;
3064 3309 2 |
3065 3310 2 | +
3066 3311 2 | Determine if this virtual display is already pasted to this
3067 3312 2 | pasteboard. If it is we can do the MOVE. If it isn't we'll do a
3068 3313 2 | PASTE at the specified position.
3069 3314 2 | -
3070 3315 2 | IF NOT SMG$$LOCATE_PP( .DCB, .PBCB, PP)
3071 3316 2 | THEN
3072 3317 2 |   RETURN SMG$$PASTE_VIRTUAL_DISPLAY(.DCB,.PBCB,
3073 3318 2 |     .PASTEBOARD_ROW,.PASTEBOARD_COL);
3074 3319 2 |
3075 3320 2 | +
3076 3321 2 | Set new row and column into pasting packet
3077 3322 2 | -
3078 3323 2 |   PP [PP_W_ROW]      = ..PASTEBOARD_ROW;
3079 3324 2 |   PP [PP_W_COL]      = ..PASTEBOARD_COL;
3080 3325 2 |
3081 3326 2 | +
3082 3327 2 | Recalc. occlusions.
3083 3328 2 | -
3084 3329 2 | IF NOT ( STATUS = SMG$$CHECK_OCCLUSION ( .PBCB))
3085 3330 2 | THEN
3086 3331 2 |   RETURN (.STATUS);
3087 3332 2 |
3088 3333 2 | +
3089 3334 2 | Recalculate the transformation constants needed to copy this display's

```



```

: 3090      3335 2 ! buffers into the associated window's buffers.
: 3091      3336 !
: 3092      3337 IF NOT ( STATUS = SMG$$CALC_PASTE_TRANSF (.PP))
: 3093      3338 THEN
: 3094      3339 RETURN (.STATUS);
: 3095      3340
: 3096      3341 RETURN (SMG$$CHECK_FOR_OUTPUT_PBCB (.PBCB));
: 3097      3342
: 3098      3343 END;
                                ! Routine SMG$MOVE_VIRTUAL_DISPLAY

```

			001C 00000	.ENTRY	SMG\$MOVE_VIRTUAL_DISPLAY, Save R2,R3,R4	3222
	54	00000000'	EF 9E 00002	MOVAB	PBD_L COUNT, R4	
	5E		04 C2 00009	SUBL2	#4, SP	
	04		6C 91 0000C	CMPB	(AP), #4	3292
			08 13 0000F	BEQL	1\$	
	50	00000000G	8F D0 00011	MOVL	#SMG\$_WRONUMARG, R0	
			04 00018	RET		
	50	04	BC D0 00019	MOVL	@DISPLAY_ID, R0	3299
04	BC	38	A0 D1 0001D	CMPL	56(R0), @DISPLAY_ID	
			06 12 00022	BNEQ	2\$	
	11	44	A0 91 00024	CMPB	68(R0), #17	
			08 13 00028	BEQL	3\$	
	50	00000000G	8F D0 0002A	MOVL	#SMG\$_INVDIS_ID, R0	
			04 00031	RET		
	52	04	BC D0 00032	MOVL	@DISPLAY_ID, DCB	
	50	08	BC D0 00036	MOVL	@PASTEBOARD_ID, R0	3300
			0A 19 0003A	BLSS	4\$	
	64		50 D1 0003C	CMPL	R0, PBD_L_COUNT	
			05 14 0003F	BGTR	4\$	
08	44	A4	50 E0 00041	BBS	R0, PBD V PB_AVAIL, 5\$	
	50	00000000G	8F D0 00046	MOVL	#SMG\$_INVPAID_ID, R0	
			04 0004D	RET		
	53	04 A440	D0 0004E	MOVL	PBD_A PBCB[R0], PBCB	
		1C	A2 D5 00053	TSTL	28(DCB)	3306
			08 13 00056	BEQL	6\$	
	50	00000000G	8F D0 00058	MOVL	#SMG\$_ILLBATFNC, R0	3308
			04 0005F	RET		
		400C	8F BB 00060	PUSHR	#*M<R2,R3,SP>	3315
0000V	CF		03 FB 00064	CALLS	#3, SMG\$\$LOCATE_PP	
	0C		50 E8 00069	BLBS	R0, 7\$	
	7E	0C	AC 7D 0006C	MOVQ	PASTEBOARD_ROW, -(SP)	3318
			0C BB 00070	PUSHR	#*M<R2,R3>	3317
0000V	CF		04 FB 00072	CALLS	#4, SMG\$\$PASTE_VIRTUAL_DISPLAY	
			04 00077	RET		
	52		6E D0 00078	MOVL	PP, R2	3323
18	A2	0C	BC B0 0007B	MOVW	@PASTEBOARD_ROW, 24(R2)	
1A	A2	10	BC B0 00080	MOVW	@PASTEBOARD_COL, 26(R2)	3324
			53 DD 00085	PUSHL	PBCB	3329
0000V	CF		01 FB 00087	CALLS	#1, SMG\$\$CHECK_OCCLUSION	
	13		50 E9 0008C	BLBC	STATUS, 8\$	
			52 DD 0008F	PUSHL	R2	3337
0000V	CF		01 FB 00091	CALLS	#1, SMG\$\$CALC_PASTE_TRANSF	
	09		50 E9 00096	BLBC	STATUS, 8\$	



SMG\$DISPLAY\_LIN 1-096 SMG\$DISPLAY LINKS - Virtual Display Linkages 16-Sep-1984 00:29:22 VAX-11 Bliss-32 V4.0-742  
SMG\$MOVE\_VIRTUAL\_DISPLAY - Move previously past 14-Sep-1984 13:09:43 [SMGRTL.SRC]SMGDISLIN.B32;1

Page 89  
(17)

00000000G 00 53 DD 00099 PUSHL PBCB  
01 FB 0009B CALLS #1, SMG\$\$CHECK\_FOR\_OUTPUT\_PBCB  
04 000A2 8\$: RET ; 3341  
; 3343

; Routine Size: 163 bytes, Routine Base: \_SMG\$CODE + 10EA

; 3099 3344 1 !<BLF/PAGE>



```

3101 3345 1 %SBTTL 'SMG$PASTE_VIRTUAL_DISPLAY - Paste virtual display to pasteboard'
3102 3346 1 GLOBAL ROUTINE SMG$PASTE_VIRTUAL_DISPLAY (
3103 3347 1     DISPLAY_ID,
3104 3348 1     PASTEBOARD_ID,
3105 3349 1     PASTEBOARD_ROW,
3106 3350 1     PASTEBOARD_COL
3107 3351 1 ) =
3108 3352 1
3109 3353 1 ++
3110 3354 1 FUNCTIONAL DESCRIPTION:
3111 3355 1     The specified virtual display is "pasted" (oriented
3112 3356 1     with respect to) a pasteboard. This makes the display visible.
3113 3357 1
3114 3358 1 CALLING SEQUENCE:
3115 3359 1
3116 3360 1     ret_status.wlc.v = SMG$PASTE_VIRTUAL_DISPLAY (
3117 3361 1     DISPLAY_ID.rl.r,
3118 3362 1     PASTEBOARD_ID.rl.r,
3119 3363 1     PASTEBOARD_ROW.rl.r,
3120 3364 1     PASTEBOARD_COL.rl.r)
3121 3365 1
3122 3366 1 FORMAL PARAMETERS:
3123 3367 1
3124 3368 1     DISPLAY_ID.rl.r      Id of virtual display to be pasted.
3125 3369 1
3126 3370 1     PASTEBOARD_ID.rl.r   The pasteboard id of the pasteboard on
3127 3371 1     which the pasting is to take place.
3128 3372 1
3129 3373 1     PASTEBOARD_ROW.rl.r  Row on pasteboard which is to contain
3130 3374 1     row 1 of the specified virtual display.
3131 3375 1
3132 3376 1     PASTEBOARD_COL.rl.r  Column on pasteboard which is to contain
3133 3377 1     column 1 of the specified virtual
3134 3378 1     display.
3135 3379 1
3136 3380 1 IMPLICIT INPUTS:
3137 3381 1
3138 3382 1     None
3139 3383 1
3140 3384 1 IMPLICIT OUTPUTS:
3141 3385 1
3142 3386 1     None
3143 3387 1
3144 3388 1 COMPLETION STATUS:
3145 3389 1
3146 3390 1     SS$ NORMAL          Normal successful completion
3147 3391 1     SMG$_INVDIS_ID      Invalid virtual display id.
3148 3392 1     SMG$_INVPAS_ID      Invalid pasteboard id.
3149 3393 1     SMG$_WRONUMARG      Wrong number of arguments.
3150 3394 1     SMG$_ILLBATFNC      Display is batched.
3151 3395 1
3152 3396 1 SIDE EFFECTS:
3153 3397 1
3154 3398 1     NONE
3155 3399 1 --
3156 3400 2 BEGIN
3157 3401 2 BUILTIN
  
```



```

3158      3402      2      AP,
3159      3403      2      CALLG;
3160      3404      2
3161      3405      2      LOCAL
3162      3406      2      STATUS,          ! Status of subroutine calls
3163      3407      2
3164      3408      2      PP      : REF $PP_DECL,      ! Addr of the pasting packet
3165      3409      2                      ! being created.
3166      3410      2      DCB      : REF $DCB_DECL,      ! Addr. of display control block
3167      3411      2      WCB      : REF $WCB_DECL,      ! Addr. of window control block
3168      3412      2      PBCB     : REF $PBCB_DECL;      ! Addr of pasteboard control
3169      3413      2                      ! block
3170      3414      2
3171      3415      2      SSMG$VALIDATE_ARGCOUNT (4, 4);      ! Test for right no. of args
3172      3416      2
3173      3417      2      +
3174      3418      2      Get addresses of associated virtual display control block and
3175      3419      2      pasteboard control block, validating both the display id and the
3176      3420      2      pasteboard id.
3177      3421      2      -
3178      3422      2      SSMG$GET_DCB ( .DISPLAY_ID, DCB);      ! Get addr of DCB
3179      3423      2      SSMG$GET_PBCB ( .PASTEBOARD_ID, PBCB);      ! Get addr of PBCB
3180      3424      2
3181      3425      2      +
3182      3426      2      Give an error if the display is batched.
3183      3427      2      -
3184      3428      2
3185      3429      2      IF .DCB[DCB_L_BATCH_LEVEL] NEQ 0
3186      3430      2      THEN
3187      3431      2      RETURN SMG$_ILLBATFNC;
3188      3432      2
3189      3433      2      +
3190      3434      2      Check to make sure we're don't already have a pasting from this
3191      3435      2      virtual display to this pasteboard. If it is, we employ the
3192      3436      2      repaste logic to remove the current pasting before allowing this new
3193      3437      2      pasting. This is necessary because we don't want ambiguous pastings.
3194      3438      2      Note: The repaste logic ends up recalling the paste routine
3195      3439      2      recursively (after doing an unpaste) -- but that's ok since there
3196      3440      2      can be at most one such pasting. The second time we are called this
3197      3441      2      test will fail.
3198      3442      2      -
3199      3443      2      IF SMG$$LOCATE_PP( .DCB, .PBCB, PP)
3200      3444      2      THEN
3201      3445      2      RETURN (CALLG (.AP, SMG$REPASTE_VIRTUAL_DISPLAY));
3202      3446      2
3203      3447      2      RETURN SMG$$PASTE_VIRTUAL_DISPLAY(.DCB,.PBCB,
3204      3448      2      .PASTEBOARD_ROW,.PASTEBOARD_COL);
3205      3449      2
3206      3450      1      END;          ! Routine SMG$PASTE_VIRTUAL_DISPLAY
  
```

```

54 00000000' 001C 00000      .ENTRY SMG$PASTE_VIRTUAL_DISPLAY, Save R2,R3,R4      : 3346
5E           EF 9E 00002      MOVAB PBD_L COUNT, R4      :
04 C2 00009      SUBL2 #4,-SP      :
  
```



SMG\$DISPLAY_LIN	SMG\$DISPLAY_LINKS - Virtual Display Linkages	16-Sep-1984 00:29:22	VAX-11 Bliss-32 V4.0-742	Page 92
1-096	SMG\$PASTE_VIRTUAL_DISPLAY - Paste virtual displ	14-Sep-1984 13:09:43	[SMGRTL.SRC]SMGDISLIN.B32;1	(18)

	04		6C	91	0000C	CMPB	(AP), #4		3415
			08	13	0000F	BEQL	1\$		
	50	00000000G	8F	D0	00011	MOVL	#SMG\$_WRONUMARG, R0		
				04	00018	RET			
04	50	04	BC	D0	00019	1\$:	MOVL	@DISPLAY_ID, R0	3422
	BC	38	A0	D1	0001D		CMPL	56(R0), @DISPLAY_ID	
			06	12	00022		BNEQ	2\$	
	11	44	A0	91	00024		CMPB	68(R0), #17	
			08	13	00028		BEQL	3\$	
	50	00000000G	8F	D0	0002A	2\$:	MOVL	#SMG\$_INVDIS_ID, R0	
				04	00031		RET		
	52	04	BC	D0	00032	3\$:	MOVL	@DISPLAY_ID, DCB	
	50	08	BC	D0	00036		MOVL	@PASTEBOARD_ID, R0	3423
			0A	19	0003A		BLSS	4\$	
	64		50	D1	0003C		CMPL	R0, PBD_L_COUNT	
			05	14	0003F		BGTR	4\$	
08	44	A4	50	E0	00041		BBS	R0, PBD V PB_AVAIL, 5\$	
			8F	D0	00046	4\$:	MOVL	#SMG\$_INVPA\$-ID, R0	
				04	0004D		RET		
	53	04	A440	D0	0004E	5\$:	MOVL	PBD A PBCB[R0], PBCB	
		1C	A2	D5	00053		TSTL	28(DCB)	3429
			08	13	00056		BEQL	6\$	
	50	00000000G	8F	D0	00058		MOVL	#SMG\$_ILLBATFNC, R0	3431
				04	0005F		RET		
		400C	8F	BB	00060	6\$:	PUSHR	#M<R2,R3,SP>	3443
0000V	CF		03	FB	00064		CALLS	#3, SMG\$\$LOCATE_PP	
	06		50	E9	00069		BLBC	R0, 7\$	
0000V	CF		6C	FA	0006C		CALLG	(AP), SMG\$REPASTE_VIRTUAL_DISPLAY	3445
				04	00071		RET		
	7E	0C	AC	7D	00072	7\$:	MOVQ	PASTEBOARD_ROW, -(SP)	3448
			0C	BB	00076		PUSHR	#M<R2,R3>	3447
0000V	CF		04	FB	00078		CALLS	#4, SMG\$\$PASTE_VIRTUAL_DISPLAY	
				04	0007D		RET		3450

; Routine Size: 126 bytes, Routine Base: \_SMG\$CODE + 118D

; 3207 3451 1 !<BLF/PAGE>



```

3209 3452 1 %SBTTL 'SMG$POP_VIRTUAL_DISPLAY - Pop off (delete) a sequence of virtual displays'
3210 3453 1 GLOBAL ROUTINE SMG$POP_VIRTUAL_DISPLAY (
3211 3454 1     DISPLAY_ID,
3212 3455 1     PASTEBOARD_ID
3213 3456 1 ) =
3214 3457 1
3215 3458 1 ++
3216 3459 1 FUNCTIONAL DESCRIPTION:
3217 3460 1     This procedure deletes all the virtual displays on the specified
3218 3461 1     pasteboard, starting with the display specified, up through all
3219 3462 1     higher-pasted display. Each of these displays is unpasted in
3220 3463 1     in the course of doing the deletion.
3221 3464 1
3222 3465 1 CALLING SEQUENCE:
3223 3466 1     ret_status.wlc.v = SMG$POP_VIRTUAL_DISPLAY ( DISPLAY_ID.rl.r,
3224 3467 1     PASTEBOARD_ID.rl.r)
3225 3468 1
3226 3469 1 FORMAL PARAMETERS:
3227 3470 1
3228 3471 1     DISPLAY_ID.rl.r     Address of the display id of the lowest
3229 3472 1                        pasted virtual display to be deleted.
3230 3473 1                        All higher-pasted displays are deleted
3231 3474 1                        as well.
3232 3475 1
3233 3476 1     PASTEBOARD_ID.rl.r Address of the pasteboard id involved.
3234 3477 1
3235 3478 1 IMPLICIT INPUTS:
3236 3479 1     NONE
3237 3480 1
3238 3481 1 IMPLICIT OUTPUTS:
3239 3482 1     NONE
3240 3483 1
3241 3484 1 COMPLETION STATUS:
3242 3485 1
3243 3486 1     SSS NORMAL      Normal successful completion
3244 3487 1     SMG$_INVDIS_ID  Invalid display id
3245 3488 1     SMG$_INVPAS_ID  Invalid pasteboard id
3246 3489 1     SMG$_WRONUMARG  Wrong number of arguments
3247 3490 1
3248 3491 1 SIDE EFFECTS:
3249 3492 1     NONE
3250 3493 1
3251 3494 1 --
3252 3495 1
3253 3496 1 BEGIN
3254 3497 1 LOCAL
3255 3498 1
3256 3499 2     STATUS,                ! Status of subr. calls
3257 3500 2
3258 3501 2     RET_STATUS,            ! Accumulated status during
3259 3502 2                           loop
3260 3503 2     PBCB : REF $PBCB_DECL, ! Address of a pasteboard
3261 3504 2                           control block
3262 3505 2
3263 3506 2     DCB : REF $DCB_DECL,   ! Address of a virtual display
3264 3507 2
3265 3508 2
  
```



```

3266      3509      2      ! control block we started with
3267      3510
3268      3511      PP : REF $PP_DECL;      ! Addr of 2 longwords that form
3269      3512      ! queue header in PP currently
3270      3513      ! under inspection.
3271      3514
3272      3515      + Check for right number of arguments.
3273      3516      -
3274      3517      SSMG$VALIDATE_ARGCOUNT ( 2, 2);
3275      3518
3276      3519      +
3277      3520      Get addresses of virtual display control block and pasteboard control
3278      3521      block and validate them.
3279      3522      -
3280      3523      SSMG$GET_PBCB ( .PASTEBOARD_ID, PBCB );
3281      3524      SSMG$GET_DCB ( .DISPLAY_ID, DCB);
3282      3525
3283      3526      +
3284      3527      Locate the pasting packet that reflects this pasting (if one exists)
3285      3528      .PP is the base address of the pasting packet.
3286      3529      -
3287      3530      IF NOT (STATUS = SMG$$LOCATE_PP ( .DCB, .PBCB, PP))
3288      3531      THEN
3289      3532      RETURN (.STATUS);
3290      3533
3291      3534      +
3292      3535      Change packet address to address of queue header.
3293      3536      -
3294      3537      PP = .PP + PP_PBCB_QUEUE_OFFSET; ! Start with specified packet
3295      3538
3296      3539      RET_STATUS = SS$_NORMAL;      ! Assume success to follow
3297      3540
3298      3541      +
3299      3542      Batch the sequence of updates we are about to do.
3300      3543      -
3301      3544      IF NOT ( STATUS = SMG$$BEGIN_PASTEBOARD_UPDATE_R1 (.PBCB))
3302      3545      THEN
3303      3546      RETURN (.STATUS);
3304      3547
3305      3548      +
3306      3549      Loop for all pasting packets starting with this one to the last-pasted
3307      3550      one...
3308      3551      -
3309      3552      WHILE .PP NEQ PBCB [PBCB_A_PP_NEXT]
3310      3553      DO
3311      3554      BEGIN      ! For all displays that need to be deleted
3312      3555      LOCAL
3313      3556      STATUS,      ! Status of delete calls
3314      3557
3315      3558      PP_BASE : REF $PP_DECL,      ! Base address of the PP
3316      3559
3317      3560      DCB : REF $DCB_DECL;      ! Current virtual display that
3318      3561      ! needs to be deleted.
3319      3562
3320      3563      +
3321      3564      Calc. the base address of this pasting packet since the queue
3322      3565      headers for this part of the chain are not at relative 0 in
      the pasting packet.
  
```



```

3323 3566 3 !-
3324 3567 3 PP_BASE = .PP - PP_PBCB_QUEUE_OFFSET;
3325 3568 3
3326 3569 3 !+
3327 3570 3 Find DCB that is in this pairing.
3328 3571 3
3329 3572 3 DCB = .PP_BASE [PP_A_DCB_ADDR];
3330 3573 3
3331 3574 3 !+
3332 3575 3 Delete this virtual display, causing it to be unpasted from
3333 3576 3 all pasteboards to which it is currently pasted.
3334 3577 3
3335 3578 3 IF NOT ( STATUS = SMG$DELETE_VIRTUAL_DISPLAY ( DCB [DCB_L_DID]))
3336 3579 3 THEN
3337 3580 3 !+
3338 3581 3 If no error yet, save this one.
3339 3582 3
3340 3583 3 BEGIN
3341 3584 3 IF .RET_STATUS THEN RET_STATUS = .STATUS;
3342 3585 3 END;
3343 3586 3
3344 3587 3 !+
3345 3588 3 Walk this chain backwards, from the packet we started with
3346 3589 3 back to the head of the chain -- since the most recently
3347 3590 3 pasted displays are at the head of the chain.
3348 3591 3
3349 3592 3 PP = .PP_BASE [PP_A_PREV_PBCB];
3350 3593 3 END; ! For all displays that need to be deleted
3351 3594 2
3352 3595 2
3353 3596 3 IF NOT (STATUS = SMG$$END_PASTEBOARD_UPDATE_R2 ( .PBCB ))
3354 3597 2 THEN
3355 3598 2 RETURN (.STATUS);
3356 3599 2
3357 3600 2 RETURN (.RET_STATUS);
3358 3601 2
3359 3602 1 END; ! End of routine SMG$POP_VIRTUAL_DISPLAY

```

		007C 00000		.ENTRY	SMG\$POP_VIRTUAL_DISPLAY, Save R2,R3,R4,R5,-	3453
					R6	
	56	00000000'	EF 9E 00002	MOVAB	PBD_L_COUNT, R6	
	5E		04 C2 00009	SUBL2	#4, SP	
	02		6C 91 0000C	CMPB	(AP), #2	3517
			08 13 0000F	BEQL	1\$	
	50	00000000G	8F D0 00011	MOVL	#SMG\$_WRONUMARG, R0	
			04 00018	RET		
	50	08	BC D0 00019 1\$:	MOVL	@PASTEBOARD_ID, R0	3523
			0A 19 0001D	BLSS	2\$	
	66		50 D1 0001F	CMPL	R0, PBD_L_COUNT	
			05 14 00022	BGTR	2\$	
08	44	A6	50 E0 00024	BBS	R0, PBD_V_PB_AVAIL, 3\$	
		50	00000000G 8F D0 00029 2\$:	MOVL	#SMG\$_INVPAS_ID, R0	
			04 00030	RET		



	55	04	A640	D0	00031	3\$:	MOVL	PBD_A_PBCB[R0], PBCB	
	50	04	BC	D0	00036		MOVL	@DISPLAY_ID, R0	3524
04	BC	38	A0	D1	0003A		CMPL	56(R0), @DISPLAY_ID	
			06	12	0003F		BNEQ	4\$	
	11	44	A0	91	00041		CMPB	68(R0), #17	
			08	13	00045		BEQL	5\$	
	50	00000000G	8F	D0	00047	4\$:	MOVL	#SMG\$_INVDIS_ID, R0	
				04	0004E		RET		
	50	04	BC	D0	0004F	5\$:	MOVL	@DISPLAY_ID, DCB	
		4021	8F	BB	00053		PUSHR	#^M<R0,R5,SP>	3530
0000V	CF		03	FB	00057		CALLS	#3, SMG\$\$LOCATE_PP	
	53		50	D0	0005C		MOVL	R0, STATUS	
	48		53	E9	0005F		BLBC	STATUS, 9\$	
	6E		08	C0	00062		ADDL2	#8, PP	3537
	54		01	D0	00065		MOVL	#1, RET_STATUS	3539
	50		55	D0	00068		MOVL	PBCB, R0	3544
		00000000G	00	16	0006B		JSB	SMG\$\$BEGIN_PASTEBOARD_UPDATE_R1	
	53		50	D0	00071		MOVL	R0, STATUS	
	33		53	E9	00074		BLBC	STATUS, 9\$	
	55		6E	D1	00077	6\$:	CMPL	PP, PBCB	3552
			1F	13	0007A		BEQL	8\$	
52	6E		08	C3	0007C		SUBL3	#8, PP, PP_BASE	3567
	50	10	A2	D0	00080		MOVL	16(PP_BASE), DCB	3572
		38	A0	9F	00084		PUSHAB	56(DCB)	3578
FB98	CF		01	FB	00087		CALLS	#1, SMG\$DELETE_VIRTUAL_DISPLAY	
	06		50	E8	0008C		BLBS	STATUS, 7\$	
	03		54	E9	0008F		BLBC	RET_STATUS, 7\$	3584
	54		50	D0	00092		MOVL	STATUS, RET_STATUS	
	6E	0C	A2	D0	00095	7\$:	MOVL	12(PP_BASE), PP	3592
			DC	11	00099		BRB	6\$	3552
	50		55	D0	0009B	8\$:	MOVL	PBCB, R0	3596
		00000000G	00	16	0009E		JSB	SMG\$\$END_PASTEBOARD_UPDATE_R2	
	53		50	D0	000A4		MOVL	R0, STATUS	
	04		53	E8	000A7		BLBS	STATUS, 10\$	
	50		53	D0	000AA	9\$:	MOVL	STATUS, R0	3598
				04	000AD		RET		
	50		54	D0	000AE	10\$:	MOVL	RET_STATUS, R0	3600
				04	000B1		RET		3602

; Routine Size: 178 bytes, Routine Base: \_SMG\$CODE + 120B

; 3360 3603 1 !<BLF/PAGE>



```

3362 3604 1 %SBTTL 'SMG$REPASTE_VIRTUAL_DISPLAY - Repaste virtual display to pasteboard'
3363 3605 1 GLOBAL ROUTINE SMG$REPASTE_VIRTUAL_DISPLAY (
3364 3606 1     DISPLAY_ID,
3365 3607 1     PASTEBOARD_ID,
3366 3608 1     PASTEBOARD_ROW,
3367 3609 1     PASTEBOARD_COL
3368 3610 1 ) =
3369 3611 1
3370 3612 1 ++
3371 3613 1 FUNCTIONAL DESCRIPTION:
3372 3614 1     The specified virtual display is "unpasted" from the specified
3373 3615 1     pasteboard. It is then "repasted" in the new position to the
3374 3616 1     same pasteboard. The unpasting and repasting operation is done
3375 3617 1     under cover of a SMG$$BEGIN PASTEBOARD_UPDATE_R1 and
3376 3618 1     SMG$$END PASTEBOARD_UPDATE_R2 pair so that there is no effect on the
3377 3619 1     screen while it is going on. Only the completed results of the
3378 3620 1     operation become visible.
3379 3621 1
3380 3622 1 CALLING SEQUENCE:
3381 3623 1
3382 3624 1     ret_status.wlc.v = SMG$REPASTE_VIRTUAL_DISPLAY (
3383 3625 1         DISPLAY_ID.rl.r,
3384 3626 1         PASTEBOARD_ID.rl.r,
3385 3627 1         PASTEBOARD_ROW.rl.r,
3386 3628 1         PASTEBOARD_COL.rl.r)
3387 3629 1
3388 3630 1 FORMAL PARAMETERS:
3389 3631 1
3390 3632 1     DISPLAY_ID.rl.r      Id of virtual display to be repasted.
3391 3633 1
3392 3634 1     PASTEBOARD_ID.rl.r   The pasteboard id of the pasteboard on
3393 3635 1                           which the unpasting/pasting is to take
3394 3636 1                           place.
3395 3637 1
3396 3638 1     PASTEBOARD_ROW.rl.r  Row on pasteboard which is to contain
3397 3639 1                           row 1 of the specified virtual display
3398 3640 1                           after repasting.
3399 3641 1
3400 3642 1     PASTEBOARD_COL.rl.r  Column on pasteboard which is to contain
3401 3643 1                           column 1 of the specified virtual
3402 3644 1                           display after repasting.
3403 3645 1
3404 3646 1 IMPLICIT INPUTS:
3405 3647 1
3406 3648 1     None
3407 3649 1
3408 3650 1 IMPLICIT OUTPUTS:
3409 3651 1
3410 3652 1     None
3411 3653 1
3412 3654 1 COMPLETION STATUS:
3413 3655 1
3414 3656 1     $$$ NORMAL          Normal successful completion
3415 3657 1     SMG$_INVDIS_ID      Invalid virtual display id.
3416 3658 1     SMG$_INVPAS_ID      Invalid pasteboard id.
3417 3659 1     SMG$_WRONUMARG      Wrong number of arguments.
3418 3660 1

```



```

3419 3661 1 SIDE EFFECTS:
3420 3662 1
3421 3663 1 NONE
3422 3664 1 --
3423 3665 2 BEGIN
3424 3666 2
3425 3667 2 LOCAL
3426 3668 2 DCB : REF $DCB_DECL,
3427 3669 2 PBCB : REF $PBCB_DECL,
3428 3670 2 STATUS ; ! Status of subroutine calls
3429 3671 2
3430 3672 2 $SMG$VALIDATE_ARGCOUNT (4, 4); ! Test for right no. of args
3431 3673 2
3432 3674 2 $SMG$GET_PBCB(.PASTEBOARD_ID,PBCB);
3433 3675 2 $SMG$GET_DCB(.DISPLAY_ID,DCB);
3434 3676 2
3435 3677 2 +
3436 3678 2 Set up an extra level of output inhibiting so that our UNPASTE
3437 3679 2 operation won't find its way to the screen until we're done.
3438 3680 2 -
3439 3681 2 IF NOT (STATUS = SMG$$BEGIN_PASTEBOARD_UPDATE_R1 (.PBCB))
3440 3682 2 THEN
3441 3683 2 RETURN (.STATUS);
3442 3684 2
3443 3685 2 +
3444 3686 2 Unpaste it from where it is.
3445 3687 2 -
3446 3688 2 IF NOT (STATUS = SMG$$UNPASTE_VIRTUAL_DISPLAY (.DCB, .PBCB))
3447 3689 2 THEN
3448 3690 2 BEGIN
3449 3691 2 SMG$$END_PASTEBOARD_UPDATE_R2 (.PBCB); ! Reduce buffering level
3450 3692 2 RETURN (.STATUS); ! Return error
3451 3693 2 END;
3452 3694 2
3453 3695 2 +
3454 3696 2 Now repaste to the same pasteboard in a new position.
3455 3697 2 -
3456 3698 2 STATUS = SMG$$PASTE_VIRTUAL_DISPLAY(.DCB,.PBCB,
3457 3699 2 .PASTEBOARD_ROW,.PASTEBOARD_COL);
3458 3700 2
3459 3701 2 +
3460 3702 2 Undo one buffering level so that we are back where we started.
3461 3703 2 -
3462 3704 2 SMG$$END_PASTEBOARD_UPDATE_R2 (.PBCB);
3463 3705 2
3464 3706 2 +
3465 3707 2 If last PASTE operation yielded an error, return that status, else
3466 3708 2 return $$$_NORMAL;
3467 3709 2 -
3468 3710 2 IF NOT .STATUS THEN RETURN .STATUS;
3469 3711 2
3470 3712 2 RETURN ($$_NORMAL);
3471 3713 2
3472 3714 1 END; ! Routine SMG$REPASTE_VIRTUAL_DISPLAY

```



			00FC 00000		.ENTRY	SMG\$REPASTE_VIRTUAL_DISPLAY, Save R2,R3,R4,-;	
						R5,R6,R7	3605
					MOVAB	SMG\$END_PASTEBOARD_UPDATE_R2, R7	
					MOVAB	PBD_L_COUNT, R6	
					CMPB	(APT, #4	3672
					BEQL	1\$	
					MOVL	#SMG\$_WRONUMARG, R0	
					RET		
					MOVL	@PASTEBOARD_ID, R0	3674
					BLSS	2\$	
					CMPL	R0, PBD_L_COUNT	
					BGTR	2\$	
					BBS	R0, PBD V PB_AVAIL, 3\$	
					MOVL	#SMG\$_INVPAS_ID, R0	
					RET		
					MOVL	PBD A PBCB[R0], PBCB	
					MOVL	@DISPLAY_ID, R0	3675
					CMPL	56(R0), @DISPLAY_ID	
					BNEQ	4\$	
					CMPB	68(R0), #17	
					BEQL	5\$	
					MOVL	#SMG\$_INVDIS_ID, R0	
					RET		
					MOVL	@DISPLAY_ID, DCB	
					MOVL	PBCB, R0	3681
					JSB	SMG\$BEGIN_PASTEBOARD_UPDATE_R1	
					MOVL	R0, STATUS	
					BLBC	STATUS, 7\$	
					PUSHL	PBCB	3688
					PUSHL	DCB	
					CALLS	#2, SMG\$UNPASTE_VIRTUAL_DISPLAY	
					MOVL	R0, STATUS	
					BLBS	STATUS, 6\$	
					MOVL	PBCB, R0	3691
					JSB	SMG\$END_PASTEBOARD_UPDATE_R2	
					BRB	7\$	3692
					MOVQ	PASTEBOARD_ROW, -(SP)	3699
					PUSHL	PBCB	3698
					PUSHL	DCB	
					CALLS	#4, SMG\$PASTE_VIRTUAL_DISPLAY	
					MOVL	R0, STATUS	
					MOVL	PBCB, R0	3704
					JSB	SMG\$END_PASTEBOARD_UPDATE_R2	
					BLBS	STATUS, 8\$	3710
					MOVL	STATUS, R0	
					RET		
					MOVL	#1, R0	3712
					RET		3714

; Routine Size: 156 bytes, Routine Base: \_SMG\$CODE + 12BD

; 3473 3715 1 !&lt;BLF/PAGE&gt;



```

3475 3716 1 %SBTTL 'SMG$RESTORE PHYSICAL SCREEN - Restore physical screen'
3476 3717 1 GLOBAL ROUTINE SMG$RESTORE_PHYSICAL_SCREEN (
3477 3718 1     PASTEBOARD_ID,
3478 3719 1     DISPLAY_ID
3479 3720 1 ) =
3480 3721 1
3481 3722 1 ++
3482 3723 1 FUNCTIONAL DESCRIPTION:
3483 3724 1
3484 3725 1     This routine reverses the effect of SMG$SAVE_PHYSICAL_SCREEN,
3485 3726 1     thereby putting the physical screen back to the point it was
3486 3727 1     at just prior to the call to SMG$SAVE_PHYSICAL_SCREEN.
3487 3728 1     The display id returned by SMG$SAVE_PHYSICAL_SCREEN must be
3488 3729 1     passed to this routine to allow the restoration to happen.
3489 3730 1
3490 3731 1 CALLING SEQUENCE:
3491 3732 1
3492 3733 1     ret_status.wlc.v = SMG$RESTORE_PHYSICAL_SCREEN (
3493 3734 1         PASTEBOARD_ID.rl.r,
3494 3735 1         DISPLAY_ID.rl.r)
3495 3736 1
3496 3737 1 FORMAL PARAMETERS:
3497 3738 1
3498 3739 1     PASTEBOARD_ID.rl.r    Address of a pasteboard id which is to
3499 3740 1                          be "restored".
3500 3741 1
3501 3742 1     DISPLAY_ID.rl.r      Returned display id invented to
3502 3743 1                          perform requested function.
3503 3744 1                          This must be the display id returned
3504 3745 1                          by SMG$SAVE_PHYSICAL_SCREEN.
3505 3746 1
3506 3747 1 IMPLICIT INPUTS:
3507 3748 1
3508 3749 1     NONE
3509 3750 1
3510 3751 1 IMPLICIT OUTPUTS:
3511 3752 1
3512 3753 1     NONE
3513 3754 1
3514 3755 1 COMPLETION STATUS:
3515 3756 1
3516 3757 1     $$$ NORMAL          Normal successful completion
3517 3758 1     SMG$_INVDIS_ID      Invalid Display Id.
3518 3759 1     SMG$_INVPAS_ID      Invalid Pasteboard Id.
3519 3760 1
3520 3761 1 SIDE EFFECTS:
3521 3762 1
3522 3763 1     NONE
3523 3764 1
3524 3765 1 --
3525 3766 1 BEGIN
3526 3767 1 LOCAL
3527 3768 2     DCB : REF $DCB_DECL, ! Address of virtual display control
3528 3769 2                      ! block involved.
3529 3770 2
3530 3771 2     PBCB : REF $PBCB_DECL, ! Address of pasteboard control block
3531 3772 2

```



```

3532 3773 2      PP : REF $PP_DECL,      ! Address of the pasting packet that
3533 3774 2      ! joins the virtual display to the
3534 3775 2      ! pasteboard.
3535 3776 2
3536 3777 2      STATUS;                ! Status of subr. calls
3537 3778 2
3538 3779 2      !+
3539 3780 2      ! Validate number of arguments.
3540 3781 2      !-
3541 3782 2      $SMG$VALIDATE_ARGCOUNT( 2,2);
3542 3783 2
3543 3784 2      !+
3544 3785 2      ! Map pasteboard id into a PBCB address, and display id into a DCB addr.
3545 3786 2      !-
3546 3787 2      $SMG$GET_PBCB ( .PASTEBOARD_ID, PBCB);
3547 3788 2      $SMG$GET_DCB  ( .DISPLAY_ID,   DCB);
3548 3789 2
3549 3790 2      !+
3550 3791 2      ! Locate the pasting packet that joins this virtual display with this
3551 3792 2      ! pasteboard.
3552 3793 2      !-
3553 3794 2      IF NOT (STATUS = SMG$$LOCATE_PP ( .DCB, .PBCB, PP))
3554 3795 2      THEN
3555 3796 2      RETURN (.STATUS);
3556 3797 2
3557 3798 2      !+
3558 3799 2      ! Invalidate our knowledge of where the physical scrolling region is on
3559 3800 2      ! the screen, since we don't know where the non_SMG user may have left
3560 3801 2      ! it.
3561 3802 2      !-
3562 3803 2      PBCB [PBCB_W_TOP_SCROLL_LINE] = 0;
3563 3804 2      PBCB [PBCB_W_BOT_SCROLL_LINE] = 0;
3564 3805 2
3565 3806 2      !+
3566 3807 2      ! Determine best way to clear affected area. If the whole screen is
3567 3808 2      ! involved we erase the whole screen in one operation. If only part
3568 3809 2      ! of the screen is involved, we have to do it a line at a time.
3569 3810 2      !-
3570 3811 2      IF .PP [PP_W_FIRST_WCB_ROW] LEQ 1      AND
3571 3812 2      .PP [PP_W_LAST_WCB_ROW] GEQ .PBCB [PBCB_B_ROWS]
3572 3813 2      THEN
3573 3814 2      BEGIN      ! Full screen involved
3574 3815 2      !+
3575 3816 2      ! Clear the whole physical screen to get rid of what the non-SMG
3576 3817 2      ! user may have put there.
3577 3818 2      !-
3578 3819 2      IF NOT (STATUS = SMG$$ERASE_PASTEBOARD (.PBCB))
3579 3820 2      THEN
3580 3821 2      RETURN (.STATUS);
3581 3822 2
3582 3823 2      END      ! Full screen involved
3583 3824 2
3584 3825 2      ELSE
3585 3826 2
3586 3827 2      BEGIN      ! Only part of screen involved
3587 3828 2      !+
3588 3829 2      ! Clear only the part of the screen involved. We'll have to do

```



```

3589 3830 3      ! it line by line.
3590 3831 3      ! The code to do that should really reside in module SMGMINUPD
3591 3832 3      ! for modularity. However, it is here for now.
3592 3833 3      -
3593 3834 3      LOCAL
3594 3835 3      WCB : REF $WCB_DECL;      ! Addr of window control block
3595 3836 3      ! involved.
3596 3837 3
3597 3838 3      WCB = .PBCB [PBCB_A_WCB];
3598 3839 3
3599 3840 3      !+
3600 3841 3      ! For each line involved, set cursor to column 1 of that line
3601 3842 3      ! and emit erase sequence. Setting the cursor to column 1 of
3602 3843 3      ! the line is necessary for non-VT100 terminals.
3603 3844 3      -
3604 3845 3
3605 3846 3      INCR I FROM .PP [PP_W_FIRST_WCB_ROW] TO .PP [PP_W_LAST_WCB_ROW]
3606 3847 3      DO
3607 3848 3      BEGIN      ! Row by row
3608 3849 3      !+
3609 3850 3      ! Set cursor to column 1 of row .I.
3610 3851 3      -
3611 3852 3      SMG$$FIND_MIN_CURSOR_POS (
3612 3853 3      .PBCB,
3613 3854 3      .WCB [WCB_W_OLD_CUR_ROW],      ! Current row
3614 3855 3      .WCB [WCB_W_OLD_CUR_COL],      ! Current col
3615 3856 3      1,      ! Desired row
3616 3857 3      1);      ! Desired col
3617 3858 3
3618 3859 3      !+
3619 3860 3      ! Get escape sequence needed to erase a line.
3620 3861 3      ! (Can't move this outside the loop since data is left
3621 3862 3      ! in memory that FIND_MIN_CURSOR_POS might touch.
3622 3863 3      -
3623 3864 3
3624 3865 3      $SMG$GET_TERM_DATA(ERASE_WHOLE_LINE);
3625 3866 3
3626 3867 3      !+
3627 3868 3      ! Erase the Ith line.
3628 3869 3      -
3629 3870 3
3630 3871 3      IF NOT (STATUS = SMG$$OUTPUT ( .PBCB,
3631 3872 3      .PBCB[PBCB_L_CAP_LENGTH],
3632 3873 3      .PBCB[PBCB_A_CAP_BUFFER]))
3633 3874 3      THEN
3634 3875 3      RETURN (.STATUS);
3635 3876 3
3636 3877 3      END;      ! Row by row
3637 3878 3      END;      ! Only part of screen involved
3638 3879 3
3639 3880 3      !+
3640 3881 3      ! Pop off the virtual display that SMG$SAVE_PHYSICAL_SCREEN placed on
3641 3882 3      ! top to cover everything up.
3642 3883 3      -
3643 3884 3      IF NOT (STATUS = SMG$POP_VIRTUAL_DISPLAY ( .DISPLAY_ID,
3644 3885 3      .PASTEBBOARD_ID))
3645 3886 3      THEN

```



```

: 3646      3887 2      RETURN (.STATUS);
: 3647      3888 2
: 3648      3889 2      RETURN (SS$_NORMAL);
: 3649      3890 2
: 3650      3891 1      END;                      ! End of routine SMG$RESTORE_PHYSICAL_SCREEN

```

			03FC 00000	.ENTRY	SMG\$RESTORE_PHYSICAL_SCREEN, Save R2,R3,R4,-;	
	59	00000000'	EF 9E 00002	MOVAB	R5,R6,R7,R8,R9	3717
	5E		14 C2 00009	SUBL2	PBD_L_COUNT, R9	
	02		6C 91 0000C	CMPB	#20, SP	3782
			08 13 0000F	BEQL	(AP), #2	
	50	00000000G	8F D0 00011	BEQL	1\$	
				MOVL	#SMG\$_WRONUMARG, R0	
				RET		
	50	04	BC D0 00019 1\$:	MOVL	@PASTEBOARD_ID, R0	3787
			0A 19 0001D	BLSS	2\$	
	69		50 D1 0001F	CMPB	R0, PBD_L_COUNT	
			05 14 00022	BGTR	2\$	
08	44	A9	50 E0 00024	BBS	R0, PBD V PB_AVAIL, 3\$	
	50	00000000G	8F D0 00029 2\$:	MOVL	#SMG\$_INVPAS_ID, R0	
				RET		
	53	04 A940	D0 00031 3\$:	MOVL	PBD A PBCB[R0], PBCB	
	50	08 BC	D0 00036	MOVL	@DISPLAY_ID, R0	3788
	08	BC 38	A0 D1 0003A	CMPB	56(R0), @DISPLAY_ID	
			06 12 0003F	BNEQ	4\$	
	11	44 A0	91 00041	CMPB	68(R0), #17	
			08 13 00045	BEQL	5\$	
	50	00000000G	8F D0 00047 4\$:	MOVL	#SMG\$_INVDIS_ID, R0	
				RET		
	50	08 BC	D0 0004F 5\$:	MOVL	@DISPLAY_ID, DCB	
		04 AE	9F 00053	PUSHAB	PP	3794
			09 BB 00056	PUSHR	#M<R0,R3>	
	0000V	CF	03 FB 00058	CALLS	#3, SMG\$\$_LOCATE_PP	
		56	50 D0 0005D	MOVL	R0, STATUS	
		27	56 E9 00060	BLBC	STATUS, 6\$	
		00F4	C3 D4 00063	CLRL	244(PBCB)	3803
		52	04 AE D0 00067	MOVL	PP, R2	3811
		01	2F A2 B1 0006B	CMPW	47(R2), #1	
			1C 1A 0006F	BGTRU	7\$	
	50	5F A3	9A 00071	MOVZBL	95(PBCB), R0	3812
	31	A2	50 B1 00075	CMPW	R0, 49(R2)	
			12 1A 00079	BGTRU	7\$	
			53 DD 0007B	PUSHL	PBCB	3819
	00000000G	00	01 FB 0007D	CALLS	#1, SMG\$\$_ERASE_PASTEBOARD	
		56	50 D0 00084	MOVL	R0, STATUS	
		78	56 E8 00087	BLBS	STATUS, 12\$	
			0086 31 0008A 6\$:	BRW	13\$	3821
		54	08 A3 D0 0008D 7\$:	MOVL	8(PBCB), WCB	3838
		58	31 A2 3C 00091	MOVZWL	49(R2), R8	3846
		57	00FC C3 9E 00095	MOVAB	252(PBCB), R7	3865
		55	0108 C3 9E 0009A	MOVAB	264(PBCB), R5	
		52	2F A2 3C 0009F	MOVZWL	47(R2), I	3873
			52 D7 000A3	DECL	I	



			57	11	000A5	BRB	11\$	
			01	DD	000A7	PUSHL	#1	3852
			52	DD	000A9	PUSHL	I	3856
	7E	26	A4	32	000AB	CVTWL	38(WCB), -(SP)	3855
	7E	24	A4	32	000AF	CVTWL	36(WCB), -(SP)	3854
			53	DD	000B3	PUSHL	PBCB	3853
00000000G	00		05	FB	000B5	CALLS	#5, SMG\$FIND_MIN_CURSOR_POS	
			67	D5	000BC	TSTL	(R7)	3865
			04	12	000BE	BNEQ	9\$	
			65	D4	000C0	CLRL	(R5)	
			25	11	000C2	BRB	10\$	
		08	AE	D4	000C4	CLRL	INPUT_ARGS	
		08	AE	9F	000C7	PUSHAB	INPUT_ARGS	
		0104	C3	DD	000CA	PUSHL	260(PBCB)	
			55	DD	000CE	PUSHL	R5	
		0100	C3	9F	000D0	PUSHAB	256(PBCB)	
10	AE	01DB	8F	3C	000D4	MOVZWL	#475, 16(SP)	
		10	AE	9F	000DA	PUSHAB	16(SP)	
			57	DD	000DD	PUSHL	R7	
00000000G	00		06	FB	000DF	CALLS	#6, SMG\$GET_TERM_DATA	
	31		50	E9	000E6	BLBC	STATUS, 15\$	
		0104	C3	DD	000E9	PUSHL	260(PBCB)	3873
			65	DD	000ED	PUSHL	(R5)	3872
			53	DD	000EF	PUSHL	PBCB	3871
00000000G	00		03	FB	000F1	CALLS	#3, SMG\$OUTPUT	
	56		50	D0	000F8	MOVL	R0, STATUS	
	15		56	E9	000FB	BLBC	STATUS, 13\$	
A5	52		58	F3	000FE	AOBLEQ	R8, I, 8\$	3846
		04	AC	DD	00102	PUSHL	PASTEBOARD_ID	3885
		08	AC	DD	00105	PUSHL	DISPLAY_ID	3884
FDA5	CF		02	FB	00108	CALLS	#2, SMG\$POP_VIRTUAL_DISPLAY	
	56		50	D0	0010D	MOVL	R0, STATUS	
	04		56	E8	00110	BLBS	STATUS, 14\$	
	50		56	D0	00113	MOVL	STATUS, R0	3887
				04	00116	RET		
	50		01	D0	00117	MOVL	#1, R0	3889
			04	0011A	15\$:	RET		3891

; Routine Size: 283 bytes, Routine Base: \_SMG\$CODE + 1359

; 3651 3892 1 !<BLF/PAGE>



```

3653 3893 1 %SBTTL 'SMG$SAVE PHYSICAL SCREEN - Save physical screen'
3654 3894 1 GLOBAL ROUTINE SMG$SAVE_PHYSICAL_SCREEN (
3655 3895 1 PASTEBOARD_ID,
3656 3896 1 DISPLAY_ID,
3657 3897 1 DESIRED_ROW_START,
3658 3898 1 DESIRED_ROW_END
3659 3899 1 ) =
3660 3900 1
3661 3901 1 ++
3662 3902 1 FUNCTIONAL DESCRIPTION:
3663 3903 1
3664 3904 1 This routine should be called before calling a procedure which
3665 3905 1 may perform output to the screen without using the SMG$
3666 3906 1 This procedure saves the state of the screen so that it can be
3667 3907 1 restored via a later call to SMG$RESTORE_PHYSICAL_SCREEN.
3668 3908 1
3669 3909 1 This routine performs 4 functions:
3670 3910 1 It:
3671 3911 1 a). Creates a virtual display which is as wide as the
3672 3912 1 physical screen and is as high indicated by the
3673 3913 1 desired_row_start and desired_row_end.
3674 3914 1 The resulting virtual display_id is returned
3675 3915 1 to the caller.
3676 3916 1
3677 3917 1 b). Pastes this virtual display to cover the screen at a
3678 3918 1 position corresponding to column 1 of desired_row_start.
3679 3919 1
3680 3920 1 c). Set the physical cursor to (1,1) in the virtual display.
3681 3921 1 This corresponds to (desired_row_start, 1) on the
3682 3922 1 physical screen.
3683 3923 1
3684 3924 1 d). Set the physical scrolling region to be the height
3685 3925 1 of the resulting virtual display.
3686 3926 1
3687 3927 1 If either desired_row_start or desired_row_end are omitted,
3688 3928 1 the first row of the physical display and the last row of the
3689 3929 1 physical display are used, respectively, in calculating the
3690 3930 1 height of the virtual display.
3691 3931 1
3692 3932 1 The effects of this routine can be reversed by doing a
3693 3933 1 SMG$RESTORE_PHYSICAL_SCREEN (Display_id.rl.r, Pasteboard_id.rl.r),
3694 3934 1 supplying the display_id returned by this routine.
3695 3935 1
3696 3936 1 CALLING SEQUENCE:
3697 3937 1
3698 3938 1 ret_status.wlc.v = SMG$SAVE_PHYSICAL_SCREEN (
3699 3939 1 PASTEBOARD_ID.rl.r,
3700 3940 1 DISPLAY_ID.wl.r
3701 3941 1 [,DESIRED_ROW_START.rl.r]
3702 3942 1 [,DESIRED_ROW_END.rl.r])
3703 3943 1
3704 3944 1 FORMAL PARAMETERS:
3705 3945 1
3706 3946 1 PASTEBOARD_ID.rl.r Address of a pasteboard id which is to
3707 3947 1 be "saved".
3708 3948 1
3709 3949 1 DISPLAY_ID.wl.r Returned display id invented to
perform requested function.

```



```

3710 3950 1 DESIRED_ROW_START.rl.r Optional. The address of the 1st row
3711 3951 1 to be "saved". If omitted, row 1 of
3712 3952 1 the physical display is used.
3713 3953 1
3714 3954 1
3715 3955 1 DESIRED_ROW_END.rl.r Optional. The address of the last row
3716 3956 1 to be "saved". If omitted, the last
3717 3957 1 row of the physical display is used.
3718 3958 1 IMPLICIT INPUTS:
3719 3959 1
3720 3960 1 NONE
3721 3961 1
3722 3962 1 IMPLICIT OUTPUTS:
3723 3963 1
3724 3964 1 NONE
3725 3965 1
3726 3966 1 COMPLETION STATUS:
3727 3967 1
3728 3968 1 SS$_NORMAL Normal successful completion
3729 3969 1
3730 3970 1 From: SMG$CREATE VIRTUAL DISPLAY
3731 3971 1 LIB$_INSVIRMEM Insufficient virtual memory
3732 3972 1
3733 3973 1 From: SMG$PASTE VIRTUAL DISPLAY
3734 3974 1 SMG$_INVPAS_ID Invalid Pasteboard Id.
3735 3975 1
3736 3976 1
3737 3977 1 SIDE EFFECTS:
3738 3978 1
3739 3979 1 The appropriate part of the physical screen will be blanked,
3740 3980 1 scrolling region will be full height of the past to be "saved",
3741 3981 1 and cursor will be at (desired_row_start,1) on screen.
3742 3982 1 --
3743 3983 1
3744 3984 2 BEGIN
3745 3985 2 BUILTIN
3746 3986 2 NULLPARAMETER;
3747 3987 2
3748 3988 2 LOCAL
3749 3989 2 ROW1,
3750 3990 2 ROWN,
3751 3991 2 FULL_SCREEN,
3752 3992 2
3753 3993 2 PBCB : REF $PBCB_DECL,
3754 3994 2
3755 3995 2
3756 3996 2 NEW_DCB : REF $DCB_DECL,
3757 3997 2
3758 3998 2
3759 3999 2
3760 4000 2
3761 4001 2 STATUS;
3762 4002 2
3763 4003 2
3764 4004 2 + Validate number of arguments and get the PBCB that goes with the
3765 4005 2 Pasteboard id.
3766 4006 2 -

```



```

3767 4007 2    $SMG$VALIDATE_ARGCOUNT( 2,4);
3768 4008
3769 4009      $SMG$GET_PBCB ( .PASTEBOARD_ID, PBCB);
3770 4010
3771 4011      +
3772 4012      Assume full screen case and intialize accordingly
3773 4013      -
3774 4014      FULL_SCREEN = 1;    ! Assume full screen
3775 4015      ROW1 = 1;
3776 4016      ROWN = .PBCB [PBCB_B_ROWS];
3777 4017
3778 4018      +
3779 4019      See which optional parameters were supplied and re-adjust assumptions.
3780 4020      -
3781 4021      IF NOT NULLPARAMETER (DESIRED_ROW_START)
3782 4022      THEN
3783 4023          BEGIN    ! Desired_row_start specified
3784 4024              FULL_SCREEN = 0;
3785 4025              ROW1 = ..DESIRED_ROW_START;
3786 4026              END;    ! Desired_row_start specified
3787 4027
3788 4028      IF NOT NULLPARAMETER (DESIRED_ROW_END)
3789 4029      THEN
3790 4030          BEGIN    ! Desired_row_end specified
3791 4031              FULL_SCREEN = 0;
3792 4032              ROWN = ..DESIRED_ROW_END;
3793 4033              END;    ! Desired_row_end specified
3794 4034      +
3795 4035      If either of the optional row parameters were supplied, make sure
3796 4036      we got a consistant range.
3797 4037      -
3798 4038      IF NOT .FULL_SCREEN
3799 4039      THEN
3800 4040          BEGIN    ! Validity check on rows
3801 4041              IF .ROW1 LSS 1                                OR ! Start off top
3802 4042                  .ROW1 GEQ .PBCB [PBCB_B_ROWS] -1        OR ! need 2 lines to scroll
3803 4043                  .ROWN LSS 1                                OR ! End off top
3804 4044                  .ROWN GTR .PBCB [PBCB_B_ROWS]            OR ! End off bottom
3805 4045                  .ROWN - .ROW1 LSS 1                      ! Wrong order
3806 4046              THEN
3807 4047                  RETURN SMG$_INVROW;
3808 4048
3809 4049              END;    ! Validity check on rows
3810 4050
3811 4051      +
3812 4052      Create a virtual display the same width as the physical screen and
3813 4053      as high as desired.
3814 4054      -
3815 4055      IF NOT (STATUS = SMG$$CREATE_VIRTUAL_DISPLAY (
3816 4056          %REF ( .ROWN - .ROW1 +1),    ! # rows
3817 4057          %REF (.PBCB [PBCB_W_WIDTH]), ! # columns
3818 4058          NEW DCB,                      ! new disp. id
3819 4059          %REF(0),                      ! default display attr
3820 4060          %REF(0),                      ! default video attr
3821 4061          %REF(0),                      ! default alt char set
3822 4062          ))
3823 4063      THEN
  
```



```

3824 4064 2      RETURN (.STATUS);
3825 4065 2
3826 4066 2  +
3827 4067 2  Paste newly-create virtual display to (desired_row_start,1) of
3828 4068 2  pasteboard.
3829 4069 2  -
3830 4070 2      IF NOT (STATUS = SMG$PASTE_VIRTUAL_DISPLAY (
3831 4071 2          .NEW_DCB,          DCB address
3832 4072 2          .PBCB,            Pasteboard control block
3833 4073 2          ROW1,            Row
3834 4074 2          %REF (1)))      Col 1
3835 4075 2      THEN
3836 4076 2          RETURN (.STATUS);
3837 4077 2
3838 4078 2  +
3839 4079 2  Set physical scrolling region to be full height of screen.
3840 4080 2  -
3841 4081 2      IF NOT (STATUS = SMG$FORCE_SCROLL_REG ( .PBCB,      Pasteboard
3842 4082 2          .ROW1,          Top row
3843 4083 2          .ROWN))      Bottom row
3844 4084 2      THEN
3845 4085 2          RETURN (.STATUS);
3846 4086 2
3847 4087 2  +
3848 4088 2  Return id of newly-create virtual display to caller.
3849 4089 2  -
3850 4090 2      .DISPLAY_ID = .NEW_DCB;
3851 4091 2
3852 4092 2      RETURN (SS$_NORMAL);
3853 4093 2
3854 4094 2      END;
3855 4095 1      ! End of routine SMG$SAVE_PHYSICAL_SCREEN

```

			001C 00000	.ENTRY	SMG\$SAVE_PHYSICAL_SCREEN, Save R2,R3,R4	3894
	54	00000000'	EF 9E 00002	MOVAB	PBD_L_COUNT, R4	
	5E		1C C2 00009	SUBL2	#28, SP	
50	6C		02 83 0000C	SUBB3	#2, (AP), DIFF	4007
	02		50 91 00010	CMPB	DIFF, #2	
			08 1B 00013	BLEQU	1\$	
	50	00000000G	8F D0 00015	MOVL	#SMG\$_WRONUMARG, R0	
			04 0001C	RET		
	50	04	BC D0 0001D 1\$:	MOVL	@PASTEBOARD_ID, R0	4009
			0A 19 00021	BLSS	2\$	
	64		50 D1 00023	CMPL	R0, PBD_L_COUNT	
			05 14 00026	BGTR	2\$	
08	44	A4	50 E0 00028	BBS	R0, PBD V PB_AVAIL, 3\$	
		50	00000000G 8F D0 0002D 2\$:	MOVL	#SMG\$_INVPAS_ID, R0	
			04 00034	RET		
	53	04	A440 D0 00035 3\$:	MOVL	PBD_A PBCB[R0], PBCB	
			01 D0 0003A	MOVL	#1, FOLL_SCREEN	4014
	18	AE	01 D0 0003D	MOVL	#1, ROW1	4015
		52	5F A3 9A 00041	MOVZBL	95(PBCB), ROWN	4016
		03	6C 91 00045	CMPB	(AP), #3	4021



			0C	1F	00048	BLSSU	4\$	
			0C	AC	D5	0004A	TSTL	12(AP)
				07	13	0004D	BEQL	4\$
				50	D4	0004F	CLRL	FULL_SCREEN
	18	AE	0C	BC	D0	00051	MOVL	@DESIRED_ROW_START, ROW1
		04		6C	91	00056	CMPB	(AP), #4
				0B	1F	00059	BLSSU	5\$
			10	AC	D5	0005B	TSTL	16(AP)
				06	13	0005E	BEQL	5\$
				50	D4	00060	CLRL	FULL_SCREEN
	52		10	BC	D0	00062	MOVL	@DESIRED_ROW_END, ROWN
		2F		50	E8	00066	BLBS	FULL_SCREEN, 7\$
			18	AE	D5	00069	TSTL	ROW1
				22	15	0006C	BLEQ	6\$
	50		5F	A3	9A	0006E	MOVZBL	95(PBCB), R0
				50	D7	00072	DECL	R0
	50		18	AE	D1	00074	CMPL	ROW1, R0
				16	18	00078	BGEQ	6\$
				52	D5	0007A	TSTL	ROWN
				12	15	0007C	BLEQ	6\$
52	5F	A3	08	00	ED	0007E	CMPZV	#0, #8, 95(PBCB), ROWN
				0A	19	00084	BLSS	6\$
		50	18	AE	01	00086	ADDL3	#1, ROW1, R0
				52	D1	0008B	CMPL	ROWN, R0
				08	18	0008E	BGEQ	7\$
		50	00000000G	8F	D0	00090	MOVL	#SMG\$_INVROW, R0
				04	00097	RET		
			10	AE	D4	00098	CLRL	16(SP)
			10	AE	9F	0009B	PUSHAB	16(SP)
			10	AE	D4	0009E	CLRL	16(SP)
			10	AE	9F	000A1	PUSHAB	16(SP)
			10	AE	D4	000A4	CLRL	16(SP)
			10	AE	9F	000A7	PUSHAB	16(SP)
			20	AE	9F	000AA	PUSHAB	NEW_DCB
	14	AE	5A	A3	3C	000AD	MOVZWL	90(PBCB), 20(SP)
			14	AE	9F	000B2	PUSHAB	20(SP)
		50	52	AE	C3	000B5	SUBL3	ROW1, ROWN, R0
	14	AE	01	A0	9E	000BA	MOVAB	1(R0), 20(SP)
			14	AE	9F	000BF	PUSHAB	20(SP)
	0000V	CF		06	FB	000C2	CALLS	#6, SMG\$\$CREATE_VIRTUAL_DISPLAY
		30		50	E9	000C7	BLBC	STATUS, 8\$
	10	AE		01	D0	000CA	MOVL	#1, 16(SP)
			10	AE	9F	000CE	PUSHAB	16(SP)
			1C	AE	9F	000D1	PUSHAB	ROW1
				53	DD	000D4	PUSHL	PBCB
			20	AE	DD	000D6	PUSHL	NEW_DCB
	0000V	CF		04	FB	000D9	CALLS	#4, SMG\$\$PASTE_VIRTUAL_DISPLAY
		19		50	E9	000DE	BLBC	STATUS, 8\$
				52	DD	000E1	PUSHL	ROWN
			1C	AE	DD	000E3	PUSHL	ROW1
				53	DD	000E6	PUSHL	PBCB
	00000000G	00		03	FB	000E8	CALLS	#3, SMG\$\$FORCE_SCROLL_REG
		08		50	E9	000EF	BLBC	STATUS, 8\$
		50	14	AE	D0	000F2	MOVL	NEW_DCB, @DISPLAY_ID
				01	D0	000F7	MOVL	#1, R0
				04	000FA	RET		
							8\$:	



SMG\$DISPLAY\_LIN SMG\$DISPLAY\_LINKS - Virtual Display Linkages 16-Sep-1984 00:29:22  
1-096 SMG\$SAVE\_PHYSICAL\_SCREEN - Save physical screen 14-Sep-1984 13:09:43

D 4

VAX-11 Bliss-32 V4.0-742  
[SMGRTL.SRC]SMGDISLIN.B32;1

Page 110  
(22)

; Routine Size: 251 bytes, Routine Base: \_SMG\$CODE + 1474

; 3856 4096 1 !<BLF/PAGE>



```
3858 4097 1 %SBTTL 'SMG$SET_DISPLAY_SCROLL_REGION - Set scrolling region in a virtual display'
3859 4098 1 GLOBAL ROUTINE SMG$SET_DISPLAY_SCROLL_REGION (
3860 4099 1     DISPLAY_ID,
3861 4100 1     TOP_LINE_OF_REGION,
3862 4101 1     BOTTOM_LINE_OF_REGION
3863 4102 1 ) =
3864 4103 1
3865 4104 1 ++
3866 4105 1 FUNCTIONAL DESCRIPTION:
3867 4106 1     This routine sets the top and bottom lines of a 'scrolling region'
3868 4107 1     in a virtual display. The scrolling region limits are used by
3869 4108 1     output routines which scroll (SMG$PUT_WITH_SCROLL and
3870 4109 1     SMG$PUT_LINE with line advancing). If this routine is called
3871 4110 1     with only a display_id, the scrolling region defaults to the
3872 4111 1     entire display.
3873 4112 1
3874 4113 1     If a top and bottom line are passed, they must be within the
3875 4114 1     display bounds. Scrolling can not occur outside the bounds of
3876 4115 1     a display.
3877 4116 1
3878 4117 1     This routine does not change the appearance of the screen or the
3879 4118 1     cursor position.
3880 4119 1
3881 4120 1 CALLING SEQUENCE:
3882 4121 1     ret_status.wlc.v = SMG$SET_DISPLAY_SCROLL_REGION (
3883 4122 1         DISPLAY_ID.rl.r
3884 4123 1         [,TOP_LINE_OF_REGION.rl.r]
3885 4124 1         [,BOTTOM_LINE_OF_REGION.rl.r])
3886 4125 1
3887 4126 1 FORMAL PARAMETERS:
3888 4127 1
3889 4128 1     DISPLAY_ID.rl.r      Display id of desired display.
3890 4129 1
3891 4130 1     TOP_LINE_OF_REGION.rl.r Optional. The top line of a scrolling
3892 4131 1                             region. Defaults to line 1 of the display.
3893 4132 1
3894 4133 1     BOTTOM_LINE_OF_REGION.rl.r
3895 4134 1                             Optional. The bottom line of a scrolling
3896 4135 1                             region. Defaults to the bottom line of the
3897 4136 1                             display.
3898 4137 1
3899 4138 1
3900 4139 1 IMPLICIT INPUTS:
3901 4140 1
3902 4141 1     NONE
3903 4142 1
3904 4143 1 IMPLICIT OUTPUTS:
3905 4144 1
3906 4145 1     NONE
3907 4146 1
3908 4147 1
3909 4148 1 COMPLETION STATUS:
3910 4149 1
3911 4150 1     SSS NORMAL      Normal successful completion
3912 4151 1     SMG$_INVARG     Bottom line is less than or equal to top line.
3913 4152 1     SMG$_INVROW     Row number is negative or too large
3914 4153 1     SMG$_WRONUMARG  Wrong number arguments.
```



```
3915 4154 1 | SIDE EFFECTS:
3916 4155 1 |
3917 4156 1 |
3918 4157 1 | NONE
3919 4158 1 | --
3920 4159 2 | BEGIN
3921 4160 2 | BUILTIN
3922 4161 2 | NULLPARAMETER;
3923 4162 2 |
3924 4163 2 | LOCAL
3925 4164 2 | TOP_LINE, | working top line
3926 4165 2 | BOTTOM_LINE, | working bottom line
3927 4166 2 | DCB : REF $DCB_DECL; | Addr. of display control block
3928 4167 2 |
3929 4168 2 | $SMG$VALIDATE_ARGCOUNT (1,3);
3930 4169 2 |
3931 4170 2 | $SMG$GET_DCB ( .DISPLAY_ID, DCB); | Get address of display control
3932 4171 2 | | block
3933 4172 2 |
3934 4173 2 | +
3935 4174 2 | Validate optional arguments.
3936 4175 2 | --
3937 4176 2 |
3938 4177 2 | TOP_LINE = .DCB [DCB_W_ROW_START]; ! init to default
3939 4178 2 |
3940 4179 2 | IF NOT NULLPARAMETER (TOP_LINE_OF_REGION)
3941 4180 2 | THEN
3942 4181 2 | BEGIN
3943 4182 2 | IF ..TOP_LINE_OF_REGION GEQ .DCB [DCB_W_ROW_START] AND
3944 4183 2 | ..TOP_LINE_OF_REGION LEQ .DCB [DCB_W_NO_ROWS]
3945 4184 2 | THEN
3946 4185 2 | TOP_LINE = ..TOP_LINE_OF_REGION
3947 4186 2 | ELSE
3948 4187 2 | RETURN (SMG$_INVROW); ! can't be outside display
3949 4188 2 | END;
3950 4189 2 |
3951 4190 2 | BOTTOM_LINE = .DCB [DCB_W_NO_ROWS]; ! init to default
3952 4191 2 |
3953 4192 2 | IF NOT NULLPARAMETER (BOTTOM_LINE_OF_REGION)
3954 4193 2 | THEN
3955 4194 2 | BEGIN
3956 4195 2 | IF ..BOTTOM_LINE_OF_REGION GEQ .DCB [DCB_W_ROW_START] AND
3957 4196 2 | ..BOTTOM_LINE_OF_REGION LEQ .DCB [DCB_W_NO_ROWS]
3958 4197 2 | THEN
3959 4198 2 | BOTTOM_LINE = ..BOTTOM_LINE_OF_REGION
3960 4199 2 | ELSE
3961 4200 2 | RETURN (SMG$_INVROW); ! can't be outside display
3962 4201 2 | END;
3963 4202 2 |
3964 4203 2 | IF .BOTTOM_LINE LEQ .TOP_LINE
3965 4204 2 | THEN
3966 4205 2 | RETURN (SMG$_INVARG); ! can't go backwards or
3967 4206 2 | | overlap
3968 4207 2 |
3969 4208 2 | +
3970 4209 2 | If we get here, we have a valid scrolling region. Store it.
3971 4210 2 | --
```



```

: 3972      4211  2
: 3973      4212  2
: 3974      4213  2
: 3975      4214  2
: 3976      4215  2
: 3977      4216  1

```

DCB [DCB\_W\_TOP\_OF\_SCRREG] = .TOP\_LINE;  
DCB [DCB\_W\_BOTTOM\_OF\_SCRREG] = .BOTTOM\_LINE;  
RETURN (SS\$\_NORMAL);  
END; ! end of routine SMG\$SET\_DISPLAY\_SCROLL\_REGION

50	6C	01	83	00002	.ENTRY	SMG\$SET_DISPLAY_SCROLL_REGION, Save R2	4098
	02	50	91	00006	SUBB3	#1, (AP), DIFF	4168
		08	1B	00009	CMPB	DIFF, #2	
	50	8F	D0	0000B	BLEQU	1\$	
			04	00012	MOVL	#SMG\$_WRONUMARG, R0	
	50	BC	D0	00013	RET		4170
04	BC	A0	D1	00017	MOVL	@DISPLAY_ID, R0	
		06	12	0001C	CMPL	56(R0), @DISPLAY_ID	
	11	A0	91	0001E	BNEQ	2\$	
		08	13	00022	CMPB	68(R0), #17	
	50	8F	D0	00024	BEQL	3\$	
			04	0002B	MOVL	#SMG\$_INVDIS_ID, R0	
	50	BC	D0	0002C	RET		
	52	60	3C	00030	MOVL	@DISPLAY_ID, DCB	4177
	02	6C	91	00033	MOVZWL	(DCB), TOP_LINE	4179
		1A	1F	00036	CMPB	(AP), #2	
		08	AC	D5	BLSSU	4\$	
		15	13	0003B	TSTL	8(AP)	
08	BC	00	ED	0003D	BEQL	4\$	4182
		32	14	00043	CMPZV	#0, #16, (DCB), @TOP_LINE_OF_REGION	
08	BC	00	ED	00045	BGTR	5\$	4183
		29	19	0004C	CMPZV	#0, #16, 2(DCB), @TOP_LINE_OF_REGION	
	52	BC	D0	0004E	BLSS	5\$	4185
	51	A0	3C	00052	MOVL	@TOP_LINE_OF_REGION, TOP_LINE	4190
	03	6C	91	00056	MOVZWL	2(DCB), BOTTOM_LINE	4192
		24	1F	00059	CMPB	(AP), #3	
		0C	AC	D5	BLSSU	6\$	
		1F	13	0005E	TSTL	12(AP)	
0C	BC	00	ED	00060	BEQL	6\$	4195
		0F	14	00066	CMPZV	#0, #16, (DCB), @BOTTOM_LINE_OF_REGION	
0C	BC	00	ED	00068	BGTR	5\$	4196
		06	19	0006F	CMPZV	#0, #16, 2(DCB), @BOTTOM_LINE_OF_REGION	
	51	BC	D0	00071	BLSS	5\$	4198
		08	11	00075	MOVL	@BOTTOM_LINE_OF_REGION, BOTTOM_LINE	
	50	8F	D0	00077	BRB	6\$	4200
			04	0007E	MOVL	#SMG\$_INVROW, R0	
	52	51	D1	0007F	RET		4203
		08	14	00082	CMPL	BOTTOM_LINE, TOP_LINE	
	50	8F	D0	00084	BGTR	7\$	4205
			04	0008B	MOVL	#SMG\$_INVARG, R0	
48	A0	52	B0	0008C	RET		4212
4A	A0	51	B0	00090	MOVW	TOP_LINE, 72(DCB)	4213
	50	01	D0	00094	MOVW	BOTTOM_LINE, 74(DCB)	4215
			04	00097	MOVL	#1, R0	4216
					RET		



SMG\$DISPLAY\_LIN SMG\$DISPLAY\_LINKS - Virtual Display Linkages H 4  
1-096 SMG\$SET\_DISPLAY\_SCROLL\_REGION - Set scrolling r 16-Sep-1984 00:29:22  
14-Sep-1984 13:09:43

VAX-11 Bliss-32 V4.0-742  
[SMGRTL.SRC]SMGDISLIN.B32;1

Page 114  
(23)

; Routine Size: 152 bytes, Routine Base: \_SMG\$CODE + 156F

; 3978 4217 1 !<BLF/PAGE>



```
3980 4218 1 %SBTTL 'SMG$UNPASTE_VIRTUAL_DISPLAY - Unpaste virtual display from pasteboard'
3981 4219 1 GLOBAL ROUTINE SMG$UNPASTE_VIRTUAL_DISPLAY (
3982 4220 1     DISPLAY_ID,
3983 4221 1     PASTEBOARD_ID
3984 4222 1 ) =
3985 4223 1
3986 4224 1 ++
3987 4225 1 FUNCTIONAL DESCRIPTION:
3988 4226 1     The specified virtual display is "unpasted" from a pasteboard.
3989 4227 1     Unpasting does not destroy the virtual display or its contents.
3990 4228 1     It merely removes its mapping to a particular pasteboard and
3991 4229 1     hence its visibility on that pasteboard.
3992 4230 1
3993 4231 1 CALLING SEQUENCE:
3994 4232 1     ret_status.wlc.v = SMG$UNPASTE_VIRTUAL_DISPLAY (
3995 4233 1         DISPLAY_ID.rl.r,
3996 4234 1         PASTEBOARD_ID.rl.r)
3997 4235 1
3998 4236 1 FORMAL PARAMETERS:
3999 4237 1
4000 4238 1     DISPLAY_ID.rl.r     Id of virtual display to be unpasted.
4001 4239 1
4002 4240 1     PASTEBOARD_ID.rl.r  The pasteboard id of the pasteboard from
4003 4241 1                          which the unpasting is to take place.
4004 4242 1
4005 4243 1 IMPLICIT INPUTS:
4006 4244 1     None
4007 4245 1
4008 4246 1 IMPLICIT OUTPUTS:
4009 4247 1     None
4010 4248 1
4011 4249 1 COMPLETION STATUS:
4012 4250 1
4013 4251 1     SSS NORMAL          Normal successful completion
4014 4252 1     SMG$_INVDIS_ID      Invalid virtual display id.
4015 4253 1     SMG$_INVPAS_ID      Invalid pasteboard id.
4016 4254 1     SMG$_WRONUMARG      Wrong number of arguments.
4017 4255 1     SMG$_NOTPASTED      Specified virtual display is not currently
4018 4256 1                          pasted to the specified pasteboard.
4019 4257 1     SMG$_ILLBATFNC      Display is batched.
4020 4258 1
4021 4259 1
4022 4260 1 SIDE EFFECTS:
4023 4261 1     NONE
4024 4262 1
4025 4263 1 --
4026 4264 1 BEGIN
4027 4265 1 LOCAL
4028 4266 2     STATUS,
4029 4267 2
4030 4268 2     ! Status of subroutine call
4031 4269 2
4032 4270 2     DCB      : REF $DCB_DECL,
4033 4271 2     PBCB      : REF $PBCB_DECL;
4034 4272 2     ! Addr of display control block
4035 4273 2     ! Addr of pasteboard control block
4036 4274 2     $SMG$VALIDATE_ARGCOUNT (2, 2);
4036 4274 2     ! Test for right no. of args
```



```

: 4037 4275 2 !+
: 4038 4276 2 !- Get addresses of virtual display control block and pasteboard control
: 4039 4277 2 !- block and validate display id and pasteboard id.
: 4040 4278 2 !-
: 4041 4279 2 !- $SMG$GET_PBCB ( .PASTEBOARD_ID, PBCB);
: 4042 4280 2 !- Get addr of pasteboard control
: 4043 4281 2 !- block
: 4044 4282 2 !-
: 4045 4283 2 !- $SMG$GET_DCB ( .DISPLAY_ID, DCB);
: 4046 4284 2 !- Get addr of virtual display
: 4047 4285 2 !- control block
: 4048 4286 2 !-
: 4049 4287 2 !+ Give an error if the display is batched.
: 4050 4288 2 !-
: 4051 4289 2 !-
: 4052 4290 2 !- IF .DCB[DCB_L_BATCH_LEVEL] NEQ 0
: 4053 4291 2 !- THEN
: 4054 4292 2 !- RETURN SMG$_ILLBATFNC;
: 4055 4293 2 !-
: 4056 4294 2 !- RETURN SMG$$UNPASTE_VIRTUAL_DISPLAY(.DCB,.PBCB)
: 4057 4295 2 !-
: 4058 4296 1 !- END;

```

! Routine SMG\$UNPASTE\_VIRTUAL\_DISPLAY

			0004 00000	.ENTRY	SMG\$UNPASTE_VIRTUAL_DISPLAY, Save R2	4219
52	00000000'	EF 9E	00002	MOVAB	PBD_L_COUNT, R2	4273
02		6C 91	00009	CMPB	(APT, #2	
		08 13	0000C	BEQL	1\$	
50	00000000G	8F D0	0000E	MOVL	#SMG\$_WRONUMARG, R0	
		04 04	00015	RET		
50	08	BC D0	00016	1\$: MOVL	@PASTEBOARD_ID, R0	4279
		0A 19	0001A	BLSS	2\$	
62		50 D1	0001C	CMPL	R0, PBD_L_COUNT	
		05 14	0001F	BGTR	2\$	
08 44	A2	50 E0	00021	BBS	R0, PBD_V PB_AVAIL, 3\$	
		8F D0	00026	2\$: MOVL	#SMG\$_INVPAS_ID, R0	
		04 04	0002D	RET		
51	04 A240	D0 D0	0002E	3\$: MOVL	PBD_A PBCB[R0], PBCB	4283
		BC D0	00033	MOVL	@DISPLAY_ID, R0	
04	BC 38	A0 D1	00037	CMPL	56(R0), @DISPLAY_ID	
		06 12	0003C	BNEQ	4\$	
11	44	A0 91	0003E	CMPB	68(R0), #17	
		08 13	00042	BEQL	5\$	
50	00000000G	8F D0	00044	4\$: MOVL	#SMG\$_INVDIS_ID, R0	
		04 04	0004B	RET		
50	04 BC	D0 D0	0004C	5\$: MOVL	@DISPLAY_ID, DCB	4290
	1C	A0 D5	00050	TSTL	28(DCB)	
		08 13	00053	BEQL	6\$	
50	00000000G	8F D0	00055	MOVL	#SMG\$_ILLBATFNC, R0	4292
		04 04	0005C	RET		
0000V CF		03 BB	0005D	6\$: PUSHR	#^M<R0,R1>	4294
		02 FB	0005F	CALLS	#2, SMG\$\$UNPASTE_VIRTUAL_DISPLAY	4296
		04 04	00064	RET		



SMG\$DISPLAY\_LIN SMG\$DISPLAY\_LINKS - Virtual Display Linkages K 4  
1-096 SMG\$UNPASTE\_VIRTUAL\_DISPLAY - Unpaste virtual d 16-Sep-1984 00:29:22  
14-Sep-1984 13:09:43

VAX-11 Bliss-32 V4.0-742  
[SMGRTL.SRC]SMGDISLIN.B32;1

Page 117  
(24)

; Routine Size: 101 bytes, Routine Base: \_SMG\$CODE + 1607

; 4059 4297 1 !<BLF/PAGE>



```

: 4061      4298 1 %SBTTL 'SMG$$CALC_PASTE_TRANSF - Calculate pasting transformation constants'
: 4062      4299 1 GLOBAL ROUTINE SMG$$CALC_PASTE_TRANSF ( PP : REF $PP_DECL ) =
: 4063      4300 1 ++
: 4064      4301 1 FUNCTIONAL DESCRIPTION:
: 4065      4302 1
: 4066      4303 1     This procedure precalculates the constants needed to efficiently
: 4067      4304 1     copy portions of the text and attributes from the virtual
: 4068      4305 1     display buffers located in the DCB to the window buffer located
: 4069      4306 1     in the WCB.
: 4070      4307 1     This data is derived from the pasting relationship between
: 4071      4308 1     the given virtual display and the pasteboard to which it is
: 4072      4309 1     pasted. The calculated constants are stored in the pasting
: 4073      4310 1     packet that reflects this pasting.
: 4074      4311 1
: 4075      4312 1
: 4076      4313 1 CALLING SEQUENCE:
: 4077      4314 1
: 4078      4315 1     ret_status.wlc.v = SMG$$CALC_PASTE_TRANSF ( PP.mab.r)
: 4079      4316 1
: 4080      4317 1 FORMAL PARAMETERS:
: 4081      4318 1
: 4082      4319 1     PP.mab.r      Address of pasting packet.
: 4083      4320 1
: 4084      4321 1 IMPLICIT INPUTS:
: 4085      4322 1
: 4086      4323 1     NONE
: 4087      4324 1
: 4088      4325 1 IMPLICIT OUTPUTS:
: 4089      4326 1
: 4090      4327 1     NONE
: 4091      4328 1
: 4092      4329 1 COMPLETION STATUS:
: 4093      4330 1
: 4094      4331 1     $$$_NORMAL      Normal successful completion
: 4095      4332 1
: 4096      4333 1 SIDE EFFECTS:
: 4097      4334 1
: 4098      4335 1     NONE
: 4099      4336 1 --
: 4100      4337 1
: 4101      4338 2 BEGIN
: 4102      4339 2 LOCAL
: 4103      4340 2     TEMP      : BLOCK [8,BYTE],      ! Temporary representation of
: 4104      4341 2                                     ! display buffer area as
: 4105      4342 2                                     ! projected on window buffer.
: 4106      4343 2     DCB      : REF $DCB_DECL,      ! Addr of DCB involved.
: 4107      4344 2     WCB      : REF $WCB_DECL,      ! Addr of WCB involved.
: 4108      4345 2     PBCB     : REF $PBCB_DECL,      ! Addr of PBCB involved.
: 4109      4346 2     OVERLAP : BLOCK [8,BYTE];      ! Describes area of overlap
: 4110      4347 2                                     ! between virtual display and
: 4111      4348 2                                     ! window buffer.
: 4112      4349 2
: 4113      4350 2     PBCB = .PP [PP_A_PBCB_ADDR] ;
: 4114      4351 2     WCB = .PBCB [PBCB_A_WCB] ;
: 4115      4352 2     DCB = .PP [PP_A_DCB_ADDR] ;
: 4116      4353 2
: 4117      4354 2 !+

```



```

4118 4355 2 ! Mark the border label as being invisible until it proves otherwise.
4119 4356 2 !-
4120 4357 2 PP [PP_W_LABEL_BYTES_TO_MOVE] = 0;
4121 4358 2 PP [PP_W_SRC_LABEL_OFF] = 0;
4122 4359 2 PP [PP_W_DST_LABEL_OFF] = 0;
4123 4360 2
4124 4361 2
4125 4362 2 TEMP [DCB_W_ROW_START] = .DCB [DCB_W_ROW_START] + .PP [PP_W_ROW]-1;
4126 4363 2 TEMP [DCB_W_NO_ROWS] = .DCB [DCB_W_NO_ROWS];
4127 4364 2 TEMP [DCB_W_COL_START] = .DCB [DCB_W_COL_START] + .PP [PP_W_COL]-1;
4128 4365 2 TEMP [DCB_W_NO_COLS] = .DCB [DCB_W_NO_COLS];
4129 4366 2
4130 4367 2 +
4131 4368 2 Check to see what part (if any) of this virtual display maps onto
4132 4369 2 the viewable part of the pasteboard -- i.e., the area that goes into
4133 4370 2 the window control block buffer.
4134 4371 2 !-
4135 4372 2 IF NOT SMG$OCCLUDE (
4136 4373 2     WCB [WCB_Q_COORD],      ! Area of window buffer
4137 4374 2     TEMP,                  ! Area of display buffer
4138 4375 2     OVERLAP )              ! Area of overlap
4139 4376 2                               ! (if any)
4140 4377 2 THEN
4141 4378 2     BEGIN ! No overlap
4142 4379 2     PP [PP_W_ROWS_TO_MOVE] = 0 ; ! There are no rows to move
4143 4380 2     +
4144 4381 2     ! If the display isn't visible, the border label isn't visible
4145 4382 2     either.      **** Not really true -- clean this up later ****
4146 4383 2     -
4147 4384 2     END ! No overlap
4148 4385 2 ELSE
4149 4386 2     BEGIN ! Overlap
4150 4387 2     LOCAL
4151 4388 2     DCB_START_ROW,      ! 1st row of display buffer that lands
4152 4389 2     in window buffer.
4153 4390 2     DCB_START_COL;      ! 1st column of display buffer that
4154 4391 2     lands in window buffer.
4155 4392 2
4156 4393 2     PP [PP_W_ROWS_TO_MOVE] = .OVERLAP [DCB_W_NO_ROWS];
4157 4394 2     PP [PP_W_MOVE_LENGTH] = .OVERLAP [DCB_W_NO_COLS];
4158 4395 2
4159 4396 2     PP [PP_W_FIRST_WCB_ROW] = .OVERLAP [DCB_W_ROW_START];
4160 4397 2     PP [PP_W_LAST_WCB_ROW] = .OVERLAP [DCB_W_ROW_START] +
4161 4398 2     .OVERLAP [DCB_W_NO_ROWS] - 1;
4162 4399 2
4163 4400 2     PP [PP_W_FIRST_WCB_COL] = .OVERLAP [DCB_W_COL_START];
4164 4401 2     PP [PP_W_LAST_WCB_COL] = .OVERLAP [DCB_W_COL_START] +
4165 4402 2     .OVERLAP [DCB_W_NO_COLS] - 1;
4166 4403 2
4167 4404 2     PP [PP_L_MOVE_SIZE] = .OVERLAP [DCB_W_NO_ROWS] *
4168 4405 2     .OVERLAP [DCB_W_NO_COLS];
4169 4406 2
4170 4407 2     DCB_START_ROW = .OVERLAP [DCB_W_ROW_START] - .PP [PP_W_ROW] + 1;
4171 4408 2     DCB_START_COL = .OVERLAP [DCB_W_COL_START] - .PP [PP_W_COL] + 1;
4172 4409 2
4173 4410 2     PP [PP_W_FROM_INDEX] = (.DCB_START_ROW - 1) * .DCB [DCB_W_NO_COLS]
4174 4411 2     + .DCB_START_COL - 1;

```



```

4175      4412  3
4176      4413  3
4177      4414  3
4178      4415  3
4179      4416  3
4180      4417  3
4181      4418  3
4182      4419  4
4183      4420  4
4184      4421  4
4185      4422  4
4186      4423  4
4187      4424  4
4188      4425  4
4189      4426  4
4190      4427  4
4191      4428  4
4192      4429  4
4193      4430  4
4194      4431  4
4195      4432  4
4196      4433  4
4197      4434  4
4198      4435  4
4199      4436  4
4200      4437  4
4201      4438  4
4202      4439  4
4203      4440  4
4204      4441  4
4205      4442  4
4206      4443  5
4207      4444  5
4208      4445  5
4209      4446  5
4210      4447  6
4211      4448  6
4212      4449  6
4213      4450  7
4214      4451  7
4215      4452  7
4216      4453  7
4217      4454  7
4218      4455  7
4219      4456  7
4220      4457  7
4221      4458  7
4222      4459  8
4223      4460  8
4224      4461  8
4225      4462  8
4226      4463  8
4227      4464  9
4228      4465  8
4229      4466  8
4230      4467  8
4231      4468  8

      PP [PP_W_TO_INDEX] = (.OVERLAP [DCB_W_ROW_START] -1) *
                           .WCB [WCB_W_NO_COLS] +
                           .OVERLAP [DCB_W_COL_START] -1;

      IF .DCB [DCB_V_BORDERED]
      THEN
      BEGIN                ! Bordered display
      LOCAL
      UPPER_ROW,           ! Row above top row of pasted display
      LOWER_ROW,           ! Row below bottom row of pasted display
      LEFT_COL,            ! Col. to left of pasted display
      RIGHT_COL,           ! Col. to right of pasted display
      LDES : REF BLOCK [,BYTE]; ! Address of dynamic descr. in
                           ! DCB that points to label
                           ! string.

      LDES = DCB [DCB_Q_LABEL_DESC];

      !+
      ! Compute the row and column numbers where the borders fall.
      ! Note these rows and columns may not map into the buffer
      ! and need to be validated before use.
      !-
      UPPER_ROW = .PP [PP_W_ROW] - 1 ;
      LOWER_ROW = .PP [PP_W_ROW] + .DCB [DCB_W_NO_ROWS];
      LEFT_COL = .PP [PP_W_COL] - 1 ;
      RIGHT_COL = .PP [PP_W_COL] + .DCB [DCB_W_NO_COLS];

      IF .LDES [DSC$W_LENGTH] NEQ 0
      THEN
      BEGIN                ! Label position computation
      CASE .DCB [DCB_B_LABEL_POS] FROM SMG$K_TOP TO SMG$K_RIGHT OF
      SET
      [SMG$K_TOP]:
      BEGIN                ! Label in top row
      IF .UPPER_ROW GEQ 1
      THEN
      BEGIN                ! Top row in buffer
      LOCAL
      DCOLS : SIGNED; ! Dest. col. start

      DCOLS =
      .PP [PP_W_COL] +
      .DCB [DCB_W_LABEL_UNITS] -2 ;

      IF .DCOLS LEQ .WCB [WCB_W_NO_COLS]
      THEN
      BEGIN                ! partially on screen
      LOCAL
      DCOLE : SIGNED; ! Dest. col end

      DCOLE = MIN ( (.LDES [DSC$W_LENGTH] + .DCOLS -1),
      (.PP [PP_W_COL] +
      .DCB [DCB_W_NO_COLS] ),
      .WCB [WCB_W_NO_COLS]);

      PP [PP_W_LABEL_BYTES_TO_MOVE] =

```



```

: 4232      4469 8
: 4233      4470 8
: 4234      4471 8
: 4235      4472 8
: 4236      4473 8
: 4237      4474 9
: 4238      4475 9
: 4239      4476 9
: 4240      4477 9
: 4241      4478 9
: 4242      4479 8
: 4243      4480 9
: 4244      4481 9
: 4245      4482 8
: 4246      4483 8
: 4247      4484 8
: 4248      4485 8
: 4249      4486 8
: 4250      4487 7
: 4251      4488 7
: 4252      4489 6
: 4253      4490 5
: 4254      4491 5
: 4255      4492 5
: 4256      4493 6
: 4257      4494 6
: 4258      4495 6
: 4259      4496 7
: 4260      4497 7
: 4261      4498 7
: 4262      4499 7
: 4263      4500 7
: 4264      4501 7
: 4265      4502 7
: 4266      4503 7
: 4267      4504 7
: 4268      4505 8
: 4269      4506 8
: 4270      4507 8
: 4271      4508 8
: 4272      4509 8
: 4273      4510 9
: 4274      4511 8
: 4275      4512 8
: 4276      4513 8
: 4277      4514 8
: 4278      4515 8
: 4279      4516 8
: 4280      4517 8
: 4281      4518 8
: 4282      4519 8
: 4283      4520 8
: 4284      4521 9
: 4285      4522 9
: 4286      4523 9
: 4287      4524 9
: 4288      4525 9

MAX ( 0, .DCOLE +1 -
MAX (0, .DCOLS) );

IF .PP [PP_W_COL] LEQ 0
THEN
  BEGIN ! Using tail end of label
    PP [PP_W_SRC_LABEL_OFF] =
      .LDES [DSC$W_LENGTH] -
      .PP [PP_W_LABEL_BYTES_TO_MOVE];
  END ! Using tail end of label
ELSE
  BEGIN ! Using front end of label
    PP [PP_W_SRC_LABEL_OFF] = 0;
  END; ! Using front end of label

  PP [PP_W_DST_LABEL_OFF] = (.UPPER_ROW -1) *
    .WCB [WCB_W_NO_COLS] +
    MAX(0, .DCOLS - 1);
  END; ! Partially on screen

END; ! Top row in buffer
END; ! Label in top row

[SMG$K BOTTOM]:
BEGIN ! Label in bottom row
IF .LOWER_ROW LEQ .WCB [WCB_W_NO_ROWS]
THEN
  BEGIN ! Bottom row in buffer
    LOCAL
      DCOLS : SIGNED; ! Dest. col. start

    DCOLS = .PP [PP_W_COL] +
      .DCB [DCB_W_LABEL_UNITS] - 2;

    IF .DCOLS LEQ .WCB [WCB_W_NO_COLS]
    THEN
      BEGIN ! Partially visible
        LOCAL
          DCOLE : SIGNED; ! Dest. col. end

          DCOLE = MIN ( (.LDES [DSC$W_LENGTH] + .DCOLS -1),
            (.PP [PP_W_COL] +
              .DCB [DCB_W_NO_COLS] ),
            .WCB [WCB_W_NO_COLS]);

          PP [PP_W_LABEL_BYTES_TO_MOVE] =
            MAX ( 0, .DCOLE + 1 -
              MAX (0, .DCOLS) );

          IF .PP [PP_W_COL] LEQ 0
          THEN
            BEGIN ! Using tail end of label
              PP [PP_W_SRC_LABEL_OFF] =
                .LDES [DSC$W_LENGTH] -
                .PP [PP_W_LABEL_BYTES_TO_MOVE];
            END ! Using tail end of label
          ELSE
            BEGIN ! Using front end of label
              PP [PP_W_SRC_LABEL_OFF] = 0;
            END;
          END;
        END;
      END;
    END;
  END;
END;

```



```

: 4289      4526 8
: 4290      4527 9
: 4291      4528 9
: 4292      4529 8
: 4293      4530 8
: 4294      4531 8
: 4295      4532 8
: 4296      4533 8
: 4297      4534 7
: 4298      4535 6
: 4299      4536 5
: 4300      4537 5
: 4301      4538 5
: 4302      4539 6
: 4303      4540 6
: 4304      4541 6
: 4305      4542 7
: 4306      4543 7
: 4307      4544 7
: 4308      4545 7
: 4309      4546 7
: 4310      4547 7
: 4311      4548 7
: 4312      4549 7
: 4313      4550 7
: 4314      4551 8
: 4315      4552 8
: 4316      4553 8
: 4317      4554 8
: 4318      4555 8
: 4319      4556 9
: 4320      4557 8
: 4321      4558 8
: 4322      4559 8
: 4323      4560 8
: 4324      4561 8
: 4325      4562 8
: 4326      4563 8
: 4327      4564 8
: 4328      4565 8
: 4329      4566 9
: 4330      4567 9
: 4331      4568 9
: 4332      4569 9
: 4333      4570 9
: 4334      4571 8
: 4335      4572 9
: 4336      4573 9
: 4337      4574 8
: 4338      4575 8
: 4339      4576 8
: 4340      4577 8
: 4341      4578 8
: 4342      4579 7
: 4343      4580 6
: 4344      4581 5
: 4345      4582 5

ELSE
  BEGIN ! Using front end of label
    PP [PP_W_SRC_LABEL_OFF] = 0;
  END; ! Using front end of label
  PP [PP_W_DST_LABEL_OFF] = (.LOWER_ROW - 1) *
    .WCB [WCB_W_NO_COLS] +
    MAX(0, .DCOLS - 1);

  END ;! Partially visible
END; ! Bottom row in buffer
END; ! Label in bottom row

[SMG$K_LEFT]:
BEGIN ! Label in left column
  IF .LEFT_COL GEQ 1
  THEN
    BEGIN ! Left column in buffer
      LOCAL
        DROWS : SIGNED; ! Dest. row start

        DROWS = .PP [PP_W_ROW] +
          .DCB [DCB_W_LABEL_UNITS] - 2;

        IF .DROWS LEQ .WCB [WCB_W_NO_ROWS]
        THEN
          BEGIN ! Partially visible
            LOCAL
              DROWE : SIGNED ; ! Dest. row end

              DROWE = MIN ( (.LDES [DSC$W_LENGTH] + .DROWS - 1),
                (.PP [PP_W_ROW] +
                  .DCB [DCB_W_NO_ROWS] ),
                .WCB [WCB_W_NO_ROWS]);

              PP [PP_W_LABEL_BYTES_TO_MOVE] =
                MAX ( 0, .DROWE + 1 -
                  MAX ( 0, .DROWS ) );

              IF .PP [PP_W_ROW] LEQ 0
              THEN
                BEGIN ! Using tail end of label
                  PP [PP_W_SRC_LABEL_OFF] =
                    .LDES [DSC$W_LENGTH] -
                    PP [PP_W_LABEL_BYTES_TO_MOVE];
                END ! Using tail end of label
              ELSE
                BEGIN ! Using front end of label
                  PP [PP_W_SRC_LABEL_OFF] = 0;
                END; ! Using front end of label

                PP [PP_W_DST_LABEL_OFF] = (.DROWS - 1) *
                  .WCB [WCB_W_NO_COLS] +
                  MAX(0, .LEFT_COL - 1);

                END; ! Partially visible
            END; ! Left column in buffer
          END; ! Label in left column
        
```



```

4346      4583 5      [SMG$K_RIGHT]:
4347      4584 6      BEGIN ! Label in right column
4348      4585 6      IF .RIGHT_COL LEQ .WCB [WCB_W_NO_COLS]
4349      4586 6      THEN
4350      4587 7      BEGIN ! Right column in buffer
4351      4588 7      LOCAL
4352      4589 7      DROWS : SIGNED; ! Dest. row start
4353      4590 7
4354      4591 7      DROWS = .PP [PP_W_ROW] +
4355      4592 7      .DCB [DCB_W_LABEL_UNITS] - 2;
4356      4593 7
4357      4594 7      IF .DROWS LEQ .WCB [WCB_W_NO_ROWS]
4358      4595 7      THEN
4359      4596 8      BEGIN ! Partially visible
4360      4597 8      LOCAL
4361      4598 8      DROWE : SIGNED ; ! Dest. row end
4362      4599 8
4363      4600 8      DROWE = MIN ( (.LDES [DSC$W_LENGTH] + .DROWS - 1),
4364      4601 9      (.PP [PP_W_ROW] +
4365      4602 8      .DCB [DCB_W_NO_ROWS] ),
4366      4603 8      .WCB [WCB_W_NO_ROWS]);
4367      4604 8
4368      4605 8      PP [PP_W_LABEL_BYTES_TO_MOVE] =
4369      4606 8      MAX ( 0, .DROWE + 1 -
4370      4607 8      MAX ( 0, .DROWS ) );
4371      4608 8
4372      4609 8
4373      4610 8      IF .PP [PP_W_ROW] LEQ 0
4374      4611 8      THEN
4375      4612 9      BEGIN ! Using tail end of label
4376      4613 9      PP [PP_W_SRC_LABEL_OFF] =
4377      4614 9      .LDES [DSC$W_LENGTH] -
4378      4615 9      .PP [PP_W_LABEL_BYTES_TO_MOVE];
4379      4616 9      END ! Using tail end of label
4380      4617 8      ELSE
4381      4618 9      BEGIN ! Using front end of label
4382      4619 9      PP [PP_W_SRC_LABEL_OFF] = 0;
4383      4620 8      END; ! Using front end of label
4384      4621 8
4385      4622 8      PP [PP_W_DST_LABEL_OFF] = (.DROWS - 1) *
4386      4623 8      .WCB [WCB_W_NO_COLS] +
4387      4624 8      MAX(0, .RIGHT_COL - 1);
4388      4625 8
4389      4626 7      END; ! Partially visible
4390      4627 6      END; ! Right column in buffer
4391      4628 5      END; ! Label in right column
4392      4629 5
4393      4630 5      [OUTRANGE]:
4394      4631 5      RETURN (SMG$_FATERRLIB);
4395      4632 5
4396      4633 5      TES;
4397      4634 4      END; ! Label position computation
4398      4635 3      END; ! Bordered display
4399      4636 2      END; ! Overlap
4400      4637 2
4401      4638 2      !+
4402      4639 2      ! If the virtual display width matches the window control block width,
  
```



```

: 4403      4640 2 | and the width of the data to be move during a mapping operation is
: 4404      4641 2 | this same width, then both the source area and the destination are
: 4405      4642 2 | contiguous sets of bytes and can be moved with a single CH$MOVE.
: 4406      4643 2 | If not, they have to be moved a row at a time since that is how many
: 4407      4644 2 | are piece-wise contiguous.
: 4408      4645 2 |
: 4409      4646 2 | PP [PP_V_CONTIG] = 0; ! Assume not contiguous until so proven.
: 4410      4647 2 | IF .DCB [DCB_W_NO_COLS] EQL .WCB [WCB_W_NO_COLS] AND
: 4411      4648 2 | .PP [PP_W_MOVE_LENGTH] EQL .DCB [DCB_W_NO_COLS]
: 4412      4649 2 | THEN
: 4413      4650 2 |     PP [PP_V_CONTIG] = 1; ! Contiguous
: 4414      4651 2 |
: 4415      4652 2 | RETURN (SS$_NORMAL);
: 4416      4653 1 | END; ! End of routine SMG$$CALC_PASTE_TRANSF

```

				OFFC 00000	.ENTRY	SMG\$\$CALC_PASTE_TRANSF, Save R2,R3,R4,R5,-			
				5E	2C	C2 00002	SUBL2	R6,R7,R8,R9,R10,R11	4299
				57	AC	D0 00005	MOVL	#44, SP	
				50	14	A7 D0 00009	MOVL	PP, R7	4350
				59	08	A0 D0 0000D	MOVL	20(R7), PBCB	
				52	10	A7 D0 00011	MOVL	8(PBCB), WCB	4351
				18 AE	24	A7 9E 00015	MOVL	16(R7), DCB	4352
					18	BE B4 0001A	MOVAB	36(R7), 24(SP)	4357
				14 AE	26	A7 9E 0001D	CLRW	@24(SP)	
					14	BE B4 00022	MOVAB	38(R7), 20(SP)	4358
				10 AE	28	A7 9E 00025	CLRW	@20(SP)	
					10	BE B4 0002A	MOVAB	40(R7), 16(SP)	4359
				0C AE	18	A7 9E 0002D	CLRW	@16(SP)	
				50	62	3C 00032	MOVAB	24(R7), 12(SP)	4362
				51	0C	BE 32 00035	MOVZWL	(DCB), R0	
				50	51	C0 00039	CVTL	@12(SP), R1	
24	AE			50	01	A3 0003C	ADDL2	R1, R0	
				26 AE	02	A2 B0 00041	SUBW3	#1, R0, TEMP	
				08 AE	1A	A7 9E 00046	MOVW	2(DCB), TEMP+2	4363
				50	04	A2 3C 0004B	MOVAB	26(R7), 8(SP)	4364
				51	08	BE 32 0004F	MOVZWL	4(DCB), R0	
				50	51	C0 00053	CVTL	@8(SP), R1	
28	AE			50	01	A3 00056	ADDL2	R1, R0	
				04 AE	06	A2 3C 0005B	SUBW3	#1, R0, TEMP+4	
				2A AE	04	AE B0 00060	MOVZWL	6(DCB), 4(SP)	4365
					1C	AE 9F 00065	MOVW	4(SP), TEMP+6	
					28	AE 9F 00068	PUSHAB	OVERLAP	4373
					59	DD 0006B	PUSHAB	TEMP	
				00000000G	00	03 FB 0006D	PUSHL	WCB	
				06	50	E8 00074	CALLS	#3, SMG\$\$OCCLUDE	
					1C	A7 B4 00077	BLBS	R0, 2\$	
					0258	31 0007A	CLRW	28(R7)	4379
				1C	AE	B0 0007D	BRW	42\$	4372
				22	AE	B0 00082	MOVW	OVERLAP+2, 28(R7)	4393
				2F	AE	B0 00087	MOVW	OVERLAP+6, 34(R7)	4394
				50	1C	AE B0 0008C	MOVW	OVERLAP, 47(R7)	4396
				51	1C	AE 3C 0008C	MOVZWL	OVERLAP, R0	4398
					1E	AE 3C 00090	MOVZWL	OVERLAP+2, R1	



31	A7	50	51	C0	00094	ADDL2	R1, R0	4400
		50	01	A3	00097	SUBW3	#1, R0, 49(R7)	
		53	20	AE	3C 0009C	MOVZWL	OVERLAP+4, R3	4402
		33	53	B0	000A0	MOVW	R3, 51(R7)	
		50	22	AE	3C 000A4	MOVZWL	OVERLAP+6, R0	4405
		51	FF	A340	9E 000A8	MOVAB	-1(R3)[R0], R1	
		35	51	B0	000AD	MOVW	R1, 53(R7)	
		50	1E	AE	3C 000B1	MOVZWL	OVERLAP+2, R0	4407
		51	22	AE	3C 000B5	MOVZWL	OVERLAP+6, R1	
2B	A7	50	51	C5	000B9	MULL3	R1, R0, 43(R7)	
		51	1C	AE	3C 000BE	MOVZWL	OVERLAP, R1	4408
		50	0C	BE	32 000C2	CVTL	@12(SP), R0	
		51	50	C2	000C6	SUBL2	R0, R1	4410
		50	08	BE	32 000C9	CVTL	@8(SP), R0	
	50	53	50	C3	000CD	SUBL3	R0, R3, R0	4411
		51	04	AE	C4 000D1	MULL2	4(SP), R1	
		54	8041	9E	000D5	MOVAB	(DCB START_COL)+[R1], R4	4413
		1E	A7	54	B0 000D9	MOVW	R4, 30(R7)	
		50	1C	AE	3C 000DD	MOVZWL	OVERLAP, R0	4414
		5A	50	D7	000E1	DECL	R0	
		50	06	A9	3C 000E3	MOVZWL	6(WCB), R10	4415
		51	FF	A340	9E 000EA	MULL2	R10, R0	
		20	A7	51	B0 000EF	MOVAB	-1(R3)[R0], R1	4417
		83	2F	A2	E9 000F3	MOVW	R1, 32(R7)	
		50	08	A2	9E 000F7	BLBC	47(DCB), 1\$	4429
		55	0C	BE	32 000FB	MOVAB	8(R2), LDES	
		5B	55	D7	000FF	CVTL	@12(SP), UPPER_ROW	4437
		51	0C	BE	32 00101	DECL	UPPER_ROW	
		58	02	A2	3C 00105	CVTL	@12(SP), R11	4438
		51	51	C0	00109	MOVZWL	2(DCB), R1	
		54	08	5B	D0 0010C	ADDL2	R1, R1	4439
		58	54	BE	32 0010F	MOVL	R11, LOWER_ROW	
		58	08	54	D7 00113	CVTL	@8(SP), LEFT_COL	
		53	04	BE	32 00115	DECL	LEFT_COL	4441
		56	58	AE	C0 00119	CVTL	@8(SP), R8	
		00	53	58	D0 0011D	ADDL2	4(SP), R8	4444
		0071	56	60	3C 00120	MOVL	R8, RIGHT_COL	
		0010	74	13	00123	MOVZWL	(LDES), R8	4448
		0012A	8F	00125	BEQ	12\$		
		3\$:	0012A	0012A	CASEB	49(DCB), #0, #3		
		0012A	0012A	0012A	.WORD	4\$-3\$, -		
		0012A	0012A	0012A		13\$-3\$, -		
		0012A	0012A	0012A		23\$-3\$, -		
		0012A	0012A	0012A		33\$-3\$, -		
		0012A	0012A	0012A		#SMG\$_FATERRLIB, R0		4631
		0012A	0012A	0012A		UPPER_ROW		4448
		0012A	0012A	0012A		12\$		4455
		0012A	0012A	0012A		@8(SP), R0		
		0012A	0012A	0012A		44(DCB), (SP)		
		0012A	0012A	0012A		(SP), R0		
		0012A	0012A	0012A		#2, DCOLS		4457
		0012A	0012A	0012A		DCOLS, R10		
		0012A	0012A	0012A		14\$		4463
		0012A	0012A	0012A		-1(DCOLS)[R6], R2		4464
		0012A	0012A	0012A		R2, R8		
		0012A	0012A	0012A		5\$		



		52		58	D0	0015B	5\$:	MOVL	R8, R2	:	4466
		5A		52	D1	0015E		CMPL	R2, R10	:	
				03	15	00161		BLEQ	6\$	:	
		52		5A	D0	00163		MOVL	R10, R2	:	
		58		50	D0	00166	6\$:	MOVL	DCOLS, R8	:	4470
				02	18	00169		BGEQ	7\$	:	
				58	D4	0016B		CLRL	R8	:	
		52		58	C2	0016D	7\$:	SUBL2	R8, R2	:	
				52	D6	00170		INCL	R2	:	4469
				02	18	00172		BGEQ	8\$	:	
				52	D4	00174		CLRL	R2	:	
		18	BE	52	B0	00176	8\$:	MOVW	R2, @24(SP)	:	
				08	BE	B5		TSTW	@8(SP)	:	4472
				08	14	0017D		BGTR	9\$	:	
	14	BE		18	BE	A3		SUBW3	@24(SP), R6, @20(SP)	:	4477
		56		03	11	00185		BRB	10\$	:	4472
				14	BE	B4		CLRW	@20(SP)	:	4481
				55	D7	0018A	9\$:	DECL	R5	:	4484
		55		5A	C4	0018C	10\$:	MULL2	R10, R5	:	4485
		02		50	F4	0018F		SOBGEQ	R0, 11\$	:	4486
				50	D4	00192		CLRL	R0	:	
	10	BE		55	A1	00194	11\$:	ADDW3	R0, R5, @16(SP)	:	
				63	11	00199	12\$:	BRB	22\$	:	4444
51	02	A9		10	00	ED	13\$:	CMPZV	#0, #16, 2(WCB), LOWER_ROW	:	4494
				76	19	001A1		BLSS	24\$	:	
		50		08	BE	32		CVTWL	@8(SP), R0	:	4501
		55		2C	A2	3C		MOVZWL	44(DCB), R5	:	
		50			55	C0		ADDL2	R5, R0	:	
		50			02	C2		SUBL2	#2, DCOLS	:	
		5A			50	D1		CMPL	DCOLS, R10	:	4503
					48	14		BGTR	22\$	:	
		52		FF	A0	46		MOVAB	-1(DCOLS)[R6], R2	:	4509
		58			52	D1		CMPL	R2, R8	:	4510
					03	15		BLEQ	15\$	:	
		52			58	D0		MOVL	R8, R2	:	
		5A			52	D1		CMPL	R2, R10	:	4512
					03	15		BLEQ	16\$	:	
		52			5A	D0		MOVL	R10, R2	:	
		55			50	D0		MOVL	DCOLS, R5	:	4516
					02	18		BGEQ	17\$	:	
					55	D4		CLRL	R5	:	
		52			55	C2		SUBL2	R5, R2	:	
					52	D6		INCL	R2	:	4515
					02	18		BGEQ	18\$	:	
					52	D4		CLRL	R2	:	
		18	BE	52	B0	001DB	18\$:	MOVW	R2, @24(SP)	:	
				08	BE	B5		TSTW	@8(SP)	:	4519
				08	14	001E2		BGTR	19\$	:	
	14	BE		18	BE	A3		SUBW3	@24(SP), R6, @20(SP)	:	4524
		56		03	11	001EA		BRB	20\$	:	4519
				14	BE	B4		CLRW	@20(SP)	:	4528
					51	D7		DECL	R1	:	4530
		51			5A	C4		MULL2	R10, R1	:	4531
		02			50	F4		SOBGEQ	R0, 21\$	:	4532
					50	D4		CLRL	R0	:	
	10	BE		51	50	A1		ADDW3	R0, R1, @16(SP)	:	
					69	11		BRB	32\$	:	4444



			54	D5	00200	23\$:	TSTL	LEFT_COL	4540
			65	15	00202		BLEQ	32\$	
		51	OC	BE	32	00204	CVTWL	@12(SP), R1	4547
		50	2C	A2	3C	00208	MOVZWL	44(DCB), R0	
		51		50	C0	0020C	ADDL2	R0, R1	
		50	FE	A1	9E	0020F	MOVAB	-2(R1), DROWS	
50	02	A9		00	ED	00213	CMPZV	#0, #16, 2(WCB), DROWS	4549
		10		6A	19	00219	BLSS	34\$	
		51	FF	A046	9E	0021B	MOVAB	-1(DROWS)[R6], R1	4555
		5B		51	D1	00220	CMPL	R1, R11	4556
		51		03	15	00223	BLEQ	25\$	
		51		5B	D0	00225	MOVL	R11, R1	
51	02	A9		00	ED	00228	CMPZV	#0, #16, 2(WCB), R1	4558
		10		04	18	0022E	BGEQ	26\$	
		51	02	A9	3C	00230	MOVZWL	2(WCB), R1	
		51		50	7D	00234	MOVQ	DROWS, R1	4562
				51	D5	00237	TSTL	R1	
				02	18	00239	BGEQ	27\$	
		52		51	D4	0023B	CLRL	R1	
				51	C2	0023D	SUBL2	R1, R2	
				52	D6	00240	INCL	R2	4561
				02	18	00242	BGEQ	28\$	
				52	D4	00244	CLRL	R2	
18	BE			52	B0	00246	MOVW	R2, @24(SP)	
			OC	BE	B5	0024A	TSTW	@12(SP)	4564
				08	14	0024D	BGTR	29\$	
	14	BE		18	A3	0024F	SUBW3	@24(SP), R6, @20(SP)	4569
		56		03	11	00255	BRB	30\$	4564
				14	BE	00257	CLRW	@20(SP)	4573
				50	D7	0025A	DECL	R0	4576
		50		5A	C4	0025C	MULL2	R10, R0	4577
		02		54	F4	0025F	SOBGEQ	R4, 31\$	4578
				54	D4	00262	CLRL	R4	
	10	BE		54	A1	00264	ADDW3	R4, R0, @16(SP)	
		50		6A	11	00269	BRB	42\$	4444
		5A		53	D1	0026B	CMPL	RIGHT_COL, R10	4585
				65	14	0026E	BGTR	42\$	
		51	OC	BE	32	00270	CVTWL	@12(SP), R1	4592
		50	2C	A2	3C	00274	MOVZWL	44(DCB), R0	
		51		50	C0	00278	ADDL2	R0, R1	
		50	FE	A1	9E	0027B	MOVAB	-2(R1), DROWS	
50	02	A9		00	ED	0027F	CMPZV	#0, #16, 2(WCB), DROWS	4594
		10		4E	19	00285	BLSS	42\$	
		51	FF	A046	9E	00287	MOVAB	-1(DROWS)[R6], R1	4600
		5B		51	D1	0028C	CMPL	R1, R11	4601
				03	15	0028F	BLEQ	35\$	
		51		5B	D0	00291	MOVL	R11, R1	
		10		00	ED	00294	CMPZV	#0, #16, 2(WCB), R1	4603
				04	18	0029A	BGEQ	36\$	
		51	02	A9	3C	0029C	MOVZWL	2(WCB), R1	
51	02	A9		50	7D	002A0	MOVQ	DROWS, R1	4607
				51	D5	002A3	TSTL	R1	
				02	18	002A5	BGEQ	37\$	
				51	D4	002A7	CLRL	R1	
		52		51	C2	002A9	SUBL2	R1, R2	
				52	D6	002AC	INCL	R2	4606
				02	18	002AE	BGEQ	38\$	



			52	D4	002B0		CLRL	R2			
	18	BE	52	B0	002B2	38\$:	MOVW	R2, a24(SP)			
			0C	BE	B5	002B6	TSTW	a12(SP)			4610
				08	14	002B9	BGTR	39\$			
14	BE		18	BE	A3	002BB	SUBW3	a24(SP), R6, a20(SP)			4615
				03	11	002C1	BRB	40\$			4610
			14	BE	B4	002C3	CLRW	a20(SP)			4619
				50	D7	002C6	DECL	R0			4622
		50		5A	C4	002C8	MULL2	R10, R0			4623
		02		53	F4	002CB	SOBGEQ	R3, 41\$			4624
				53	D4	002CE	CLRL	R3			
10	BE			53	A1	002D0	ADDW3	R3, R0, a16(SP)			
	2A	50		02	8A	002D5	BICB2	#2, 42(R7)			4646
	04	A7		0A	B1	002D9	CMPW	6(WCB), 4(SP)			4647
		AE	06	0B	12	002DE	BNEQ	43\$			
	04	AE	22	A7	B1	002E0	CMPW	34(R7), 4(SP)			4648
				04	12	002E5	BNEQ	43\$			
	2A	A7		02	88	002E7	BISB2	#2, 42(R7)			4650
		50		01	D0	002EB	MOVL	#1, R0			4652
				04	002EE	43\$:	RET				4653

; Routine Size: 751 bytes, Routine Base: \_SMG\$CODE + 166C

; 4417 4654 1 !<BLF/PAGE>



```

4419 4655 1 %SBTTL 'SMG$CHECK_OCCLUSION - Check pastings for occlusion'
4420 4656 1 GLOBAL ROUTINE SMG$CHECK_OCCLUSION (
4421 4657 1     PBCB : REF $PBCB_DECL
4422 4658 1     ) =
4423 4659 1
4424 4660 1 ++
4425 4661 1 FUNCTIONAL DESCRIPTION:
4426 4662 1     This procedure checks the overlap in all the pastings to
4427 4663 1     a given pasteboard -- setting a bit in the respective pasting
4428 4664 1     packets to record whether each pasting is occluded by
4429 4665 1     higher (more-recently pasted) pastings. This procedure is
4430 4666 1     invoked whenever there is a change in the pasting order or
4431 4667 1     pasting position of any virtual display.
4432 4668 1     If we can determine at pasting time that a particular virtual
4433 4669 1     display (as pasted) and with a given succession of higher-pasted
4434 4670 1     virtual displays is not occluded by any of them,
4435 4671 1     we can make mapping run faster since a change to a virtual
4436 4672 1     display which is not occluded causes only the changed virtual
4437 4673 1     display to be remapped. Higher pasted virtual displays do not
4438 4674 1     have to be remapped since it is known that they do not occlude
4439 4675 1     the changed one.
4440 4676 1
4441 4677 1 CALLING SEQUENCE:
4442 4678 1
4443 4679 1     ret_status.wlc.v = SMG$CHECK_OCCLUSION ( PBCB.rab.r)
4444 4680 1
4445 4681 1 FORMAL PARAMETERS:
4446 4682 1
4447 4683 1     PBCB.rab.r           Address of pasteboard control block
4448 4684 1                        which has something new or different
4449 4685 1                        pasted to it.
4450 4686 1
4451 4687 1 IMPLICIT INPUTS:
4452 4688 1
4453 4689 1     NONE
4454 4690 1
4455 4691 1 IMPLICIT OUTPUTS:
4456 4692 1
4457 4693 1     NONE
4458 4694 1
4459 4695 1 COMPLETION STATUS:
4460 4696 1
4461 4697 1     $$$_NORMAL          Normal successful completion
4462 4698 1
4463 4699 1 SIDE EFFECTS:
4464 4700 1
4465 4701 1     NONE
4466 4702 1
4467 4703 1 --
4468 4704 2 BEGIN
4469 4705 2 LOCAL
4470 4706 2     THIS_PP : REF $PP_DECL,
4471 4707 2             ! Addr of pasting packet for upper-most pasted
4472 4708 2             ! virtual display.
4473 4709 2
4474 4710 2     THIS_Q_HEAD : REF BLOCK [,BYTE];
4475 4711 2             ! Addr of 2 longwords that form queue header in
  
```



```

4476      4712      3      ! PP currently under inspection.
4477      4713      3
4478      4714      3
4479      4715      3      + To initialize for the rest of the algorithm, run through whole pasting
4480      4716      3      list marking all packets not occluded.
4481      4717      3      -
4482      4718      3      THIS_Q_HEAD = .PBCB [PBCB_A_PP_NEXT];      ! 1st (more recent pasting)
4483      4719      3      WHILE .THIS_Q_HEAD NEQ PBCB [PBCB_A_PP_NEXT]
4484      4720      3      DO
4485      4721      3      BEGIN      ! Init. pass
4486      4722      3      THIS_PP = .THIS_Q_HEAD - PP_PBCB_QUEUE_OFFSET;      ! To top of packet
4487      4723      3      THIS_PP [PP_V_OCCLUDED] = 0;      ! Init to not occluded
4488      4724      3      THIS_Q_HEAD = .THIS_PP [PP_A_NEXT_PBCB];      ! To queue header in
4489      4725      3      next packet
4490      4726      3      END;      ! Init. pass
4491      4727      3
4492      4728      3      THIS_Q_HEAD = .PBCB [PBCB_A_PP_NEXT];
4493      4729      3      THIS_PP = .THIS_Q_HEAD - PP_PBCB_QUEUE_OFFSET;
4494      4730      3
4495      4731      3      +
4496      4732      3      - Loop for all pasting packets starting with most-recently pasted one.
4497      4733      3      -
4498      4734      3      WHILE .THIS_Q_HEAD NEQ PBCB [PBCB_A_PP_NEXT]
4499      4735      3      DO
4500      4736      3      BEGIN      ! For all displays from top to bottom
4501      4737      3      LOCAL
4502      4738      3      NEXT_PP      : REF $PP_DECL,
4503      4739      3      ! Addr of pasting packet currently under
4504      4740      3      ! inspection.
4505      4741      3
4506      4742      3      NEXT_PP_Q_HEAD : REF BLOCK [BYTE],
4507      4743      3      ! Addr of 2 longwords that form queue
4508      4744      3      ! header in PP currently under
4509      4745      3      ! inspection.
4510      4746      3
4511      4747      3      TEMP_THIS : BLOCK [8,BYTE],
4512      4748      3      ! Area of projection of THIS virtual
4513      4749      3      ! display on pasteboard
4514      4750      3
4515      4751      3      TEMP_NEXT : BLOCK [8,BYTE],
4516      4752      3      ! Area of projection of NEXT virtual
4517      4753      3      ! display on pasteboard
4518      4754      3
4519      4755      3      THIS_DCB : REF BLOCK [BYTE];
4520      4756      3      ! Addr of virtual display currently
4521      4757      3      ! under inspection.
4522      4758      3
4523      4759      3      +
4524      4760      3      ! Recalculate pasting packet address and DCB address for this
4525      4761      3      ! iteration.
4526      4762      3      -
4527      4763      3      THIS_PP = .THIS_Q_HEAD - PP_PBCB_QUEUE_OFFSET;
4528      4764      3      THIS_DCB = .THIS_PP [PP_A_DCB_ADDR];
4529      4765      3
4530      4766      3      +
4531      4767      3      ! It is safe to assume that there is at least one virtual
4532      4768      3      ! display pasted to this pasteboard -- but there may not be more

```



```

4533 4769 3      | than one. Be careful about reaching ahead to a packet that
4534 4770 3      | may not be a packet. If doesn't exist, pointer will be
4535 4771 3      | pointing back into PBCB -- and inner loop will not be
4536 4772 3      | executed.
4537 4773 3      |
4538 4774 3      | NEXT_PP_Q_HEAD = .THIS_PP [PP_A_NEXT_PBCB];
4539 4775 3      |
4540 4776 3      | IF .NEXT_PP_Q_HEAD NEQ PBCB [PBCB_A_PP_NEXT]
4541 4777 3      | THEN
4542 4778 4      | BEGIN      ! NEXT exists
4543 4779 4      | NEXT_PP = .NEXT_PP_Q_HEAD - PP_PBCB_QUEUE_OFFSET;
4544 4780 4      | +
4545 4781 4      | Form a representation of the projection of THIS virtual
4546 4782 4      | display onto pasteboard coordinate system.
4547 4783 4      |
4548 4784 4      | TEMP_THIS [DCB_W_ROW_START] = .THIS_DCB [DCB_W_ROW_START] +
4549 4785 4      | .THIS_PP [PP_W_ROW] = 1;
4550 4786 4      | TEMP_THIS [DCB_W_NO_ROWS] = .THIS_DCB [DCB_W_NO_ROWS];
4551 4787 4      | TEMP_THIS [DCB_W_COE_START] = .THIS_DCB [DCB_W_COE_START] +
4552 4788 4      | .THIS_PP [PP_W_COL] = 1;
4553 4789 4      | TEMP_THIS [DCB_W_NO_COLS] = .THIS_DCB [DCB_W_NO_COLS];
4554 4790 4      |
4555 4791 4      | +
4556 4792 4      | If this virtual display is bordered, its projection is
4557 4793 4      | bigger than if it were not. Adjust its projection
4558 4794 4      | representation.
4559 4795 4      |
4560 4796 4      | IF .THIS_DCB [DCB_V_BORDERED]
4561 4797 4      | THEN
4562 4798 5      | BEGIN      ! Border adjustment
4563 4799 5      | TEMP_THIS [DCB_W_ROW_START] = .TEMP_THIS [DCB_W_ROW_START] - 1;
4564 4800 5      | TEMP_THIS [DCB_W_NO_ROWS] = .TEMP_THIS [DCB_W_NO_ROWS] + 2;
4565 4801 5      | TEMP_THIS [DCB_W_COE_START] = .TEMP_THIS [DCB_W_COE_START] - 1;
4566 4802 5      | TEMP_THIS [DCB_W_NO_COLS] = .TEMP_THIS [DCB_W_NO_COLS] + 2;
4567 4803 4      | END;      ! Border adjustment
4568 4804 4      |
4569 4805 3      | END;      ! Next exists
4570 4806 3      |
4571 4807 3      | WHILE .NEXT_PP_Q_HEAD NEQ PBCB [PBCB_A_PP_NEXT]
4572 4808 3      | DO
4573 4809 4      | BEGIN      ! For all displays from current to bottom
4574 4810 4      | LOCAL
4575 4811 4      | NEXT_DCB : REF $DCB DECL
4576 4812 4      |           ! Addr of DCB associated with NEXT_PP
4577 4813 4      | OVERLAP : BLOCK [8,BYTE];
4578 4814 4      |           ! Returned by SMG$OCCLUDE, but not
4579 4815 4      |           ! used in this context
4580 4816 4      |
4581 4817 4      | NEXT_PP = .NEXT_PP_Q_HEAD - PP_PBCB_QUEUE_OFFSET;
4582 4818 4      | NEXT_DCB = .NEXT_PP [PP_A_DCB_ADDR];
4583 4819 4      |
4584 4820 4      | +
4585 4821 4      | Form a representation of the projection of NEXT virtual
4586 4822 4      | display onto pasteboard coordinate system.
4587 4823 4      |
4588 4824 4      | TEMP_NEXT [DCB_W_ROW_START] = .NEXT_DCB [DCB_W_ROW_START] +
4589 4825 4      | .NEXT_PP [PP_W_ROW] = 1;

```



```

4590      4826 4      TEMP_NEXT [DCB_W_NO_ROWS] = .NEXT_DCB [DCB_W_NO_ROWS];
4591      4827 4      TEMP_NEXT [DCB_W_COE_START] = .NEXT_DCB [DCB_W_COE_START] +
4592      4828 4      .NEXT_PP [PP_W_COL] - 1;
4593      4829 4      TEMP_NEXT [DCB_W_NO_COLS] = .NEXT_DCB [DCB_W_NO_COLS];
4594      4830 4
4595      4831 4      !+
4596      4832 4      ! If this next virtual display is bordered, its projection is
4597      4833 4      ! bigger than if it were not. Adjust its projection
4598      4834 4      ! representation.
4599      4835 4      -
4600      4836 4      IF .NEXT_DCB [DCB_V_BORDERED]
4601      4837 4      THEN
4602      4838 5          BEGIN ! Border adjustment
4603      4839 5              TEMP_NEXT [DCB_W_ROW_START] = .TEMP_NEXT [DCB_W_ROW_START] - 1;
4604      4840 5              TEMP_NEXT [DCB_W_NO_ROWS] = .TEMP_NEXT [DCB_W_NO_ROWS] + 2;
4605      4841 5              TEMP_NEXT [DCB_W_COE_START] = .TEMP_NEXT [DCB_W_COE_START] - 1;
4606      4842 5              TEMP_NEXT [DCB_W_NO_COLS] = .TEMP_NEXT [DCB_W_NO_COLS] + 2;
4607      4843 4          END; ! Border adjustment
4608      4844 4
4609      4845 4      !+
4610      4846 4      ! Check to see if THIS virtual display occludes NEXT
4611      4847 4      ! virtual display and if so set occlusion bit of NEXT.
4612      4848 4      -
4613      4849 4      IF SMG$$OCCLUDE ( TEMP_NEXT, TEMP_THIS, OVERLAP)
4614      4850 4      THEN
4615      4851 4          NEXT_PP [PP_V_OCCLUDED] = 1;
4616      4852 4
4617      4853 4      !+
4618      4854 4      ! Walk chain in direction of earlier pasted packets.
4619      4855 4      -
4620      4856 4      NEXT_PP_Q_HEAD = .NEXT_PP [PP_A_NEXT_PBCB];
4621      4857 4
4622      4858 3      END; ! For all displays from current to bottom
4623      4859 3
4624      4860 3      !+
4625      4861 3      ! Walk chain in direction of earlier pasted packets.
4626      4862 3      -
4627      4863 3      THIS_Q_HEAD = .THIS_PP [PP_A_NEXT_PBCB];
4628      4864 3
4629      4865 2      END; ! For all displays from top to bottom
4630      4866 2
4631      4867 2      RETURN (SS$NORMAL);
4632      4868 2
4633      4869 1      END; ! End of routine SMG$$CHECK_OCCLUSION

```

		00FC 00000	.ENTRY	SMG\$\$CHECK_OCCLUSION, Save R2,R3,R4,R5,R6,-	4656
				R7	
5E		18 C2 00002	SUBL2	#24, SP	
56	04	AC D0 00005	MOVL	PBCB, R6	4718
54		66 D0 00009	MOVL	(R6), THIS_Q_HEAD	
56		54 D1 0000C 1\$:	CMPL	THIS_Q_HEAD, R6	4719
		0E 13 0000F	BEQL	2\$	
52	F8	A4 9E 00011	MOVAB	-8(R4), THIS_PP	4722



2A	A2	01	8A	00015	BICB2	#1, 42(THIS_PP)	4723
	54	08	A2	D0 00019	MOVL	8(THIS_PP), -THIS_Q_HEAD	4724
			ED	11 0001D	BRB	1\$	4719
	54		66	D0 0001F	2\$: MOVL	(R6), THIS_Q_HEAD	4728
	52	F8	A4	9E 00022	MOVAB	-8(R4), THIS_PP	4729
	56		54	D1 00026	3\$: CMPL	THIS_Q_HEAD, -R6	4734
			03	12 00029	BNEQ	4\$	
			00BB	31 0002B	BRW	9\$	
	52	F8	A4	9E 0002E	4\$: MOVAB	-8(R4), THIS_PP	4763
	50	10	A2	D0 00032	MOVL	16(THIS_PP), -THIS_DCB	4764
	55	08	A2	D0 00036	MOVL	8(THIS_PP), NEXT_PP_Q_HEAD	4774
	56		55	D1 0003A	CMPL	NEXT_PP_Q_HEAD, R6	4776
			3F	13 0003D	BEQL	5\$	
	53	F8	A5	9E 0003F	MOVAB	-8(R5), NEXT_PP	4779
	51		60	3C 00043	MOVZWL	(THIS_DCB), R1	4785
	57	18	A2	32 00046	CVTWL	24(THIS_PP), R7	
	51		57	C0 0004A	ADDL2	R7, R1	
10	AE		01	A3 0004D	SUBW3	#1, R1, TEMP_THIS	
	12	02	A0	B0 00052	MOVW	2(THIS_DCB), -TEMP_THIS+2	4786
		04	A0	3C 00057	MOVZWL	4(THIS_DCB), R1	4788
		1A	A2	32 0005B	CVTWL	26(THIS_PP), R7	
			57	C0 0005F	ADDL2	R7, R1	
14	AE		01	A3 00062	SUBW3	#1, R1, TEMP_THIS+4	
	16	06	A0	B0 00067	MOVW	6(THIS_DCB), -TEMP_THIS+6	4789
		2F	A0	E9 0006C	BLBC	47(THIS_DCB), 5\$	4796
		10	AE	B7 00070	DECW	TEMP_THIS	4799
	12	02	A0	00073	ADDW2	#2, TEMP_THIS+2	4800
		14	AE	B7 00077	DECW	TEMP_THIS+4	4801
	16	02	A0	0007A	ADDW2	#2, TEMP_THIS+6	4802
		55	D1	0007E	5\$: CMPL	NEXT_PP_Q_HEAD, R6	4807
	56		5F	13 00081	BEQL	8\$	
	53	F8	A5	9E 00083	MOVAB	-8(R5), NEXT_PP	4817
	50	10	A3	D0 00087	MOVL	16(NEXT_PP), -NEXT_DCB	4818
	51		60	3C 0008B	MOVZWL	(NEXT_DCB), R1	4825
	57	18	A3	32 0008E	CVTWL	24(NEXT_PP), R7	
	51		57	C0 00092	ADDL2	R7, R1	
08	AE		01	A3 00095	SUBW3	#1, R1, TEMP_NEXT	
	0A	02	A0	B0 0009A	MOVW	2(NEXT_DCB), -TEMP_NEXT+2	4826
		04	A0	3C 0009F	MOVZWL	4(NEXT_DCB), R1	4828
		1A	A3	32 000A3	CVTWL	26(NEXT_PP), R7	
			57	C0 000A7	ADDL2	R7, R1	
0C	AE		01	A3 000AA	SUBW3	#1, R1, TEMP_NEXT+4	
	0E	06	A0	B0 000AF	MOVW	6(NEXT_DCB), -TEMP_NEXT+6	4829
		2F	A0	E9 000B4	BLBC	47(NEXT_DCB), 6\$	4836
		08	AE	B7 000B8	DECW	TEMP_NEXT	4839
	0A	02	A0	000BB	ADDW2	#2, TEMP_NEXT+2	4840
		0C	AE	B7 000BF	DECW	TEMP_NEXT+4	4841
	0E	02	A0	000C2	ADDW2	#2, TEMP_NEXT+6	4842
		5E	DD	000C6	6\$: PUSHL	SP	4849
		14	AE	9F 000C8	PUSHAB	TEMP_THIS	
		10	AE	9F 000CB	PUSHAB	TEMP_NEXT	
00000000G	00		03	FB 000CE	CALLS	#3, SMG\$OCCLUDE	
	04		50	E9 000D5	BLBC	R0, 7\$	
2A	A3		01	88 000D8	BISB2	#1, 42(NEXT_PP)	4851
	55	08	A3	D0 000DC	7\$: MOVL	8(NEXT_PP), -NEXT_PP_Q_HEAD	4856
			9C	11 000E0	BRB	5\$	4807
	54	08	A2	D0 000E2	8\$: MOVL	8(THIS_PP), THIS_Q_HEAD	4863



```

SMG$DISPLAY_LIN SMG$DISPLAY_LINKS - Virtual Display Linkages      16-Sep-1984 00:29:22  VAX-11 Bliss-32 V4.0-742
1-096           SMG$$CHECK_OCCLUSION - Check pastings for occlu 14-Sep-1984 13:09:43  [SMGRTL.SRC]SMGDISLIN.B32;1

```

```

50      FF3D 31 000E6      BRW      3$
        01 D0 000E9 9$:    MOVL    #1, R0
        04 000EC      RET

```

```
; Routine Size: 237 bytes,    Routine Base: _SMG$CODE + 195B
```

SMG  
1-0



```

4636 4871 1 %SBTTL 'SMG$$CHECK_OCCLUSION_FIRST - Check pastings for occlusion'
4637 4872 1 GLOBAL ROUTINE SMG$$CHECK_OCCLUSION_FIRST (
4638 4873 1     PBCB : REF $PBCB_DECL
4639 4874 1 ) =
4640 4875 1 ++
4641 4876 1 FUNCTIONAL DESCRIPTION:
4642 4877 1
4643 4878 1     This procedure updates the overlap bit in all the pastings to
4644 4879 1     a given pasteboard -- setting a bit in the respective pasting
4645 4880 1     packets to record whether each pasting is occluded by
4646 4881 1     the highest (most-recently pasted) pasting. This procedure is
4647 4882 1     invoked whenever a virtual display is freshly pasted.
4648 4883 1
4649 4884 1     This routine differs from SMG$$CHECK_OCCLUSION it that that
4650 4885 1     routine compares all pastings to all other pasting. This
4651 4886 1     routine only compares the top pasting to all the others since
4652 4887 1     the addition of this new top one can only add an occlusion to
4653 4888 1     a lower one. Any occlusions already existing at a low level
4654 4889 1     cannot have been modified by pasting one more on top and there
4655 4890 1     is no reason to recalculate their relationship to each other.
4656 4891 1
4657 4892 1     If we can determine at pasting time that a particular virtual
4658 4893 1     display (as pasted) and with a given succession of higher-pasted
4659 4894 1     virtual displays is not occluded by any of them,
4660 4895 1     we can make mapping run faster since a change to a virtual
4661 4896 1     display which is not occluded causes only the changed virtual
4662 4897 1     display to be remapped. Higher pasted virtual displays do not
4663 4898 1     have to be remapped since it is known that they do not occlude
4664 4899 1     the changed one.
4665 4900 1
4666 4901 1 CALLING SEQUENCE:
4667 4902 1
4668 4903 1     ret_status.wlc.v = SMG$$CHECK_OCCLUSION_FIRST ( PBCB.rab.r)
4669 4904 1
4670 4905 1 FORMAL PARAMETERS:
4671 4906 1
4672 4907 1     PBCB.rab.r           Address of pasteboard control block
4673 4908 1                           which has a new virtual display pasted
4674 4909 1                           to it.
4675 4910 1
4676 4911 1 IMPLICIT INPUTS:
4677 4912 1
4678 4913 1     NONE
4679 4914 1
4680 4915 1 IMPLICIT OUTPUTS:
4681 4916 1
4682 4917 1     NONE
4683 4918 1
4684 4919 1 COMPLETION STATUS:
4685 4920 1
4686 4921 1     $$$_NORMAL           Normal successful completion
4687 4922 1
4688 4923 1 SIDE EFFECTS:
4689 4924 1
4690 4925 1     NONE
4691 4926 1 --
4692 4927 1

```



```

4693 4928 2 BEGIN
4694 4929 LOCAL
4695 4930 THIS_PP : REF $PP_DECL,
4696 4931 ! Addr of pasting packet for upper-most pasted
4697 4932 ! virtual display.
4698 4933
4699 4934 THIS_Q_HEAD : REF BLOCK [,BYTE],
4700 4935 ! Addr of 2 longwords that form queue header in
4701 4936 ! PP currently under inspection.
4702 4937
4703 4938 NEXT_PP : REF $PP_DECL,
4704 4939 ! Addr of pasting packet currently under
4705 4940 ! inspection.
4706 4941
4707 4942 NEXT_PP_Q_HEAD : REF BLOCK [,BYTE],
4708 4943 ! Addr of 2 longwords that form queue
4709 4944 ! header in PP currently under
4710 4945 ! inspection.
4711 4946
4712 4947 TEMP_THIS : BLOCK [8,BYTE],
4713 4948 ! Area of projection of THIS virtual
4714 4949 ! display on pasteboard
4715 4950
4716 4951 TEMP_NEXT : BLOCK [8,BYTE],
4717 4952 ! Area of projection of NEXT virtual
4718 4953 ! display on pasteboard
4719 4954
4720 4955 THIS_DCB : REF $DCB_DECL;
4721 4956
4722 4957 ! Addr of virtual display currently
4723 4958 ! under inspection.
4724 4959
4725 4960 THIS_Q_HEAD = .PBCB [PBCB_A_PP_NEXT]; ! Most recent pasting
4726 4961 THIS_PP = .THIS_Q_HEAD - PP_PBCB_QUEUE_OFFSET;
4727 4962
4728 4963 THIS_DCB = .THIS_PP [PP_A_DCB_ADDR];
4729 4964
4730 4965
4731 4966 + It is safe to assume that there is at least one virtual
4732 4967 display pasted to this pasteboard -- but there may not be more than
4733 4968 one. Be careful about reaching ahead to a packet that may not be a
4734 4969 packet. If doesn't exist, pointer will be pointing back into PBCB
4735 4970 -- and inner loop will not be executed.
4736 4971 -
4737 4972 NEXT_PP_Q_HEAD = .THIS_PP [PP_A_NEXT_PBCB];
4738 4973
4739 4974 IF .NEXT_PP_Q_HEAD NEQ PBCB [PBCB_A_PP_NEXT]
4740 4975 THEN
4741 4976 BEGIN ! NEXT exists
4742 4977 NEXT_PP = .NEXT_PP_Q_HEAD - PP_PBCB_QUEUE_OFFSET;
4743 4978 +
4744 4979 ! Form a representation of the projection of THIS virtual
4745 4980 ! display onto pasteboard coordinate system.
4746 4981
4747 4982 TEMP_THIS [DCB_W_ROW_START] = .THIS_DCB [DCB_W_ROW_START] +
4748 4983 .THIS_PP [PP_W_ROW] - 1;
4749 4984 TEMP_THIS [DCB_W_NO_ROWS] = .THIS_DCB [DCB_W_NO_ROWS];

```



```

4750      4985      3      TEMP_THIS [DCB_W_COL_START] = .THIS_DCB [DCB_W_COL_START] +
4751      4986      3      .THIS_PP [PP_W_COL] = 1;
4752      4987      3      TEMP_THIS [DCB_W_NO_COLS] = .THIS_DCB [DCB_W_NO_COLS];
4753      4988      3
4754      4989      3      !+
4755      4990      3      ! If this virtual display is bordered, its projection is bigger
4756      4991      3      ! than if it were not. Adjust its projection representation.
4757      4992      3      !-
4758      4993      3      IF .THIS_DCB [DCB_V_BORDERED]
4759      4994      3      THEN
4760      4995      4      BEGIN      ! Border adjustment
4761      4996      4      TEMP_THIS [DCB_W_ROW_START] = .TEMP_THIS [DCB_W_ROW_START] - 1;
4762      4997      4      TEMP_THIS [DCB_W_NO_ROWS] = .TEMP_THIS [DCB_W_NO_ROWS] + 2;
4763      4998      4      TEMP_THIS [DCB_W_COE_START] = .TEMP_THIS [DCB_W_COE_START] - 1;
4764      4999      4      TEMP_THIS [DCB_W_NO_COLS] = .TEMP_THIS [DCB_W_NO_COLS] + 2;
4765      5000      4      END;      ! Border adjustment
4766      5001      3
4767      5002      3      END;      ! Next exists
4768      5003      3
4769      5004      3      WHILE .NEXT_PP_Q_HEAD NEQ PBCB [PBCB_A_PP_NEXT]
4770      5005      3      DO
4771      5006      3      BEGIN      ! For all displays from current to bottom
4772      5007      3      LOCAL
4773      5008      3      NEXT_DCB : REF $DCB_DECL,
4774      5009      3      ! Addr of DCB associated with NEXT_PP
4775      5010      3      OVERLAP : BLOCK [8,BYTE];
4776      5011      3      ! Returned by SMG$OCCLUDE, but not
4777      5012      3      ! used in this context
4778      5013      3
4779      5014      3      NEXT_PP = .NEXT_PP_Q_HEAD - PP_PBCB_QUEUE_OFFSET;
4780      5015      3      NEXT_DCB = .NEXT_PP [PP_A_DCB_ADDR];
4781      5016      3
4782      5017      3      !+
4783      5018      3      ! Form a representation of the projection of NEXT virtual
4784      5019      3      ! display onto pasteboard coordinate system.
4785      5020      3      !-
4786      5021      3      TEMP_NEXT [DCB_W_ROW_START] = .NEXT_DCB [DCB_W_ROW_START] +
4787      5022      3      .NEXT_PP [PP_W_ROW] = 1;
4788      5023      3      TEMP_NEXT [DCB_W_NO_ROWS] = .NEXT_DCB [DCB_W_NO_ROWS];
4789      5024      3      TEMP_NEXT [DCB_W_COE_START] = .NEXT_DCB [DCB_W_COE_START] +
4790      5025      3      .NEXT_PP [PP_W_COL] = 1;
4791      5026      3      TEMP_NEXT [DCB_W_NO_COLS] = .NEXT_DCB [DCB_W_NO_COLS];
4792      5027      3
4793      5028      3      !+
4794      5029      3      ! If this next virtual display is bordered, its projection is
4795      5030      3      ! bigger than if it were not. Adjust its projection
4796      5031      3      ! representation.
4797      5032      3      !-
4798      5033      3      IF .NEXT_DCB [DCB_V_BORDERED]
4799      5034      3      THEN
4800      5035      4      BEGIN      ! Border adjustment
4801      5036      4      TEMP_NEXT [DCB_W_ROW_START] = .TEMP_NEXT [DCB_W_ROW_START] - 1;
4802      5037      4      TEMP_NEXT [DCB_W_NO_ROWS] = .TEMP_NEXT [DCB_W_NO_ROWS] + 2;
4803      5038      4      TEMP_NEXT [DCB_W_COE_START] = .TEMP_NEXT [DCB_W_COE_START] - 1;
4804      5039      4      TEMP_NEXT [DCB_W_NO_COLS] = .TEMP_NEXT [DCB_W_NO_COLS] + 2;
4805      5040      4      END;      ! Border adjustment
4806      5041      3

```



```

: 4807      5042      3      +
: 4808      5043      3      + Check to see if THIS virtual display occludes NEXT
: 4809      5044      3      + virtual display and if so set occlusion bit of NEXT.
: 4810      5045      3      -
: 4811      5046      3      IF SMG$OCCLUDE ( TEMP_NEXT, TEMP_THIS, OVERLAP)
: 4812      5047      3      THEN
: 4813      5048      3      NEXT_PP [PP_V_OCCLUDED] = 1;
: 4814      5049      3
: 4815      5050      3      +
: 4816      5051      3      + Walk chain in direction of earlier pasted packets.
: 4817      5052      3      -
: 4818      5053      3      NEXT_PP_Q_HEAD = .NEXT_PP [PP_A_NEXT_PBCB];
: 4819      5054      3
: 4820      5055      3      END;      ! For all displays from current to bottom
: 4821      5056      3
: 4822      5057      3
: 4823      5058      3      RETURN (SS$_NORMAL);
: 4824      5059      3
: 4825      5060      1      END;      ! End of routine SMG$CHECK_OCCLUSION_FIRST

```

				003C 00000	.ENTRY	SMG\$CHECK_OCCLUSION_FIRST, Save R2,R3,R4,-		
					R5		4872	
		5E	18	C2	00002	SUBL2	#24, SP	
		51	04	BC	D0 00005	MOVL	@PBCB, THIS_Q_HEAD	4960
		51		08	C2 00009	SUBL2	#8, THIS_PP	4961
		50	10	A1	D0 0000C	MOVL	16(THIS_PP), THIS_DCB	4963
		53	08	A1	D0 00010	MOVL	8(THIS_PP), NEXT_PP_Q_HEAD	4972
	04	AC		53	D1 00014	CMPL	NEXT_PP_Q_HEAD, PBCB	4974
				3F	13 00018	BEQL	1\$	
		52	F8	A3	9E 0001A	MOVAB	-8(R3), NEXT_PP	4977
		54		60	3C 0001E	MOVZWL	(THIS_DCB), R4	4983
		55	18	A1	32 00021	CVTWL	24(THIS_PP), R5	
		54		55	C0 00025	ADDL2	R5, R4	
10	AE	54		01	A3 00028	SUBW3	#1, R4, TEMP_THIS	
		12	AE	02	A0 B0 0002D	MOVW	2(THIS_DCB), -TEMP_THIS+2	4984
		54	04	A0	3C 00032	MOVZWL	4(THIS_DCB), R4	4986
		51	1A	A1	32 00036	CVTWL	26(THIS_PP), R1	
		51		54	C0 0003A	ADDL2	R4, R1	
14	AE	51		01	A3 0003D	SUBW3	#1, R1, TEMP_THIS+4	
		16	AE	06	A0 B0 00042	MOVW	6(THIS_DCB), -TEMP_THIS+6	4987
		0E		2F	A0 E9 00047	BLBC	47(THIS_DCB), 1\$	4993
				10	AE B7 0004B	DECW	TEMP_THIS	4996
		12	AE	02	A0 0004E	ADDW2	#2, TEMP_THIS+2	4997
				14	AE B7 00052	DECW	TEMP_THIS+4	4998
		16	AE	02	A0 00055	ADDW2	#2, TEMP_THIS+6	4999
		04	AC	53	D1 00059 1\$:	CMPL	NEXT_PP_Q_HEAD, PBCB	5004
				5F	13 0005D	BEQL	4\$	
		52	F8	A3	9E 0005F	MOVAB	-8(R3), NEXT_PP	5014
		50	10	A2	D0 00063	MOVL	16(NEXT_PP), -NEXT_DCB	5015
		51		60	3C 00067	MOVZWL	(NEXT_DCB), R1	5022
		54	18	A2	32 0006A	CVTWL	24(NEXT_PP), R4	
		51		54	C0 0006E	ADDL2	R4, R1	
08	AE	51		01	A3 00071	SUBW3	#1, R1, TEMP_NEXT	



	0A	AE	02	A0	B0	00076	MOVW	2(NEXT_DCB), TEMP_NEXT+2	5023
		51	04	A0	3C	0007B	MOVZWL	4(NEXT_DCB), R1	5025
		54	1A	A2	32	0007F	CVTWL	26(NEXT_PP), R4	
		51		54	C0	00083	ADDL2	R4, R1	
OC	AE	51		01	A3	00086	SUBW3	#1, R1, TEMP_NEXT+4	
	0E	AE	06	A0	B0	0008B	MOVW	6(NEXT_DCB), TEMP_NEXT+6	5026
		0E	2F	A0	E9	00090	BLBC	47(NEXT_DCB), 2\$	5033
			08	AE	B7	00094	DECW	TEMP_NEXT	5036
	0A	AE		02	A0	00097	ADDW2	#2, TEMP_NEXT+2	5037
			0C	AE	B7	0009B	DECW	TEMP_NEXT+4	5038
	0E	AE		02	A0	0009E	ADDW2	#2, TEMP_NEXT+6	5039
				5E	DD	000A2	PUSHL	SP	5046
			14	AE	9F	000A4	PUSHAB	TEMP_THIS	
			10	AE	9F	000A7	PUSHAB	TEMP_NEXT	
00000000G	00			03	FB	000AA	CALLS	#3, SMG\$\$OCCLUDE	
	04			50	E9	000B1	BLBC	R0, 3\$	
2A	A2			01	88	000B4	BISB2	#1, 42(NEXT_PP)	5048
	53		08	A2	D0	000B8	MOVL	8(NEXT_PP), NEXT_PP_Q_HEAD	5053
				9B	11	000BC	BRB	1\$	5004
	50			01	D0	000BE	MOVL	#1, R0	5058
				04	00	00C1	RET		5060

; Routine Size: 194 bytes,

Routine Base: \_SMG\$CODE + 1A48

; 4826

5061 1 !<BLF/PAGE>



```

4828 5062 1 %SBTTL 'SMG$$CREATE_PASTEBOARD - Create Pasteboard Control Block (PBCB)'
4829 5063 1 GLOBAL ROUTINE SMG$$CREATE_PASTEBOARD (
4830 5064 1     ROWS,
4831 5065 1     COLS,
4832 5066 1     PBCB_ADDR
4833 5067 1 ) =
4834 5068 1
4835 5069 1 ++
4836 5070 1 FUNCTIONAL DESCRIPTION:
4837 5071 1     This routine allocates space for a pasteboard control block.
4838 5072 1     It also allocates a buffer called the output buffer
4839 5073 1     which is used to buffer output to this terminal
4840 5074 1     (if buffering is enabled).
4841 5075 1
4842 5076 1 CALLING SEQUENCE:
4843 5077 1
4844 5078 1     ret_status.wlc.v = SMG$$CREATE_PASTEBOARD (
4845 5079 1     ROWS.rl.r,
4846 5080 1     COLS.rl.r,
4847 5081 1     PBCB.wl.r)
4848 5082 1
4849 5083 1 FORMAL PARAMETERS:
4850 5084 1
4851 5085 1     ROWS.rl.r      Max. number of rows that a window onto this
4852 5086 1                   pasteboard will have.
4853 5087 1
4854 5088 1     COLS.rl.r      Max. number of columns that a window onto this
4855 5089 1                   pasteboard will have.
4856 5090 1
4857 5091 1     PBCB_ADDR.wl.r Address of the newly-created PBCB -- returned to
4858 5092 1                   caller.
4859 5093 1
4860 5094 1 IMPLICIT INPUTS:
4861 5095 1
4862 5096 1     NONE
4863 5097 1
4864 5098 1 IMPLICIT OUTPUTS:
4865 5099 1
4866 5100 1     items in PBCB get filled in.
4867 5101 1     in particular, an output buffer is allocated.
4868 5102 1
4869 5103 1 COMPLETION STATUS:
4870 5104 1
4871 5105 1     $$$ NORMAL      Normal successful completion
4872 5106 1     LIB$_INSVIRMEM  Insufficient virtual memory to allocate needed
4873 5107 1                     buffers.
4874 5108 1
4875 5109 1 SIDE EFFECTS:
4876 5110 1
4877 5111 1     NONE
4878 5112 1 !--

```



```

: 4880      5113 2      BEGIN
: 4881      5114 2      LOCAL
: 4882      5115 2      PBCB : REF $PBCB_DECL, ! Address of PBCB allocated.
: 4883      5116 2
: 4884      5117 2      STATUS;          ! Status of subroutine calls
: 4885      5118 2
: 4886      5119 2      LITERAL
: 4887      5120 2
: 4888      5121 2      PBCB_K_OUTBUF_DEFAULT_SIZE
: 4889      5122 2
: 4890      5123 2      = 256;          ! Default size for output buffer
: 4891      5124 2      ! (if all other algorithms fail)
: 4892      5125 2
: 4893      5126 2      +
: 4894      5127 2      Allocate space for the PBCB itself.
: 4895      5128 2      -
: 4896      5129 2      IF NOT (STATUS = LIB$GET_VM (%REF (PBCB_K_SIZE), PBCB))
: 4897      5130 2      THEN
: 4898      5131 2      RETURN (.STATUS);
: 4899      5132 2
: 4900      5133 2      CH$FILL (0, PBCB_K_SIZE, .PBCB);    ! Clear all fields to default 0
: 4901      5134 2
: 4902      5135 2      +
: 4903      5136 2      Allocate the window control block that goes along with this
: 4904      5137 2      pasteboard, returning failure if we can't.
: 4905      5138 2      -
: 4906      5139 2      IF NOT (STATUS = SMG$CREATE_WCB (.ROWS, .COLS, PBCB [PBCB_A_WCB]))
: 4907      5140 2      THEN
: 4908      5141 2      BEGIN    ! No more space
: 4909      5142 2      +
: 4910      5143 2      If we can't get space for WCB, we might as well give back
: 4911      5144 2      the PBCB space itself.
: 4912      5145 2      -
: 4913      5146 2      LIB$FREE_VM (%REF (PBCB_K_SIZE), PBCB);
: 4914      5147 2      RETURN (.STATUS);
: 4915      5148 2      END;    ! No more space
: 4916      5149 2
: 4917      5150 2      +
: 4918      5151 2      Allocate output buffer that goes along with this pasteboard, returning
: 4919      5152 2      failure if we can't.
: 4920      5153 2      This buffer is used (if buffering is enabled) to buffer all output to
: 4921      5154 2      this terminal.
: 4922      5155 2      When V3B comes out, we should do a better job in figuring
: 4923      5156 2      out a good size for this buffer by looking at sysgen paramaters
: 4924      5157 2      and user quotas, etc. For now, we just allocate a fixed space.
: 4925      5158 2      -
: 4926      5159 2
: 4927      5160 2      STATUS = LIB$GET_VM (%REF (PBCB_K_OUTBUF_DEFAULT_SIZE),
: 4928      5161 2      PBCB [PBCB_A_OUTPUT_BUFFER]);
: 4929      5162 2      IF NOT .STATUS
: 4930      5163 2      THEN
: 4931      5164 2      BEGIN
: 4932      5165 2      +
: 4933      5166 2      If we can't get space for the output buffer, we might as well
: 4934      5167 2      give back the PBCB space itself as well as the WCB space.
: 4935      5168 2      Ignore any errors that occur while trying to free this space.
: 4936      5169 2      -

```



```

4937 5170 3      SMG$DEALLOCATE_WCB( .PBCB [PBCB_A_WCB] );
4938 5171 3      LIB$FREE_VM (XREF (PBCB_K_SIZE), PBCB);
4939 5172 3      RETURN (STATUS);
4940 5173 3      END;
4941 5174 2
4942 5175 2      PBCB [PBCB_W_OUTPUT_BUFSIZ] = PBCB_K_OUTBUF_DEFAULT_SIZE; ! allocation
4943 5176 2
4944 5177 2      +
4945 5178 2      Initialize pasting queue header to self.
4946 5179 2      -
4947 5180 2      PBCB [PBCB_A_PP_NEXT] = PBCB [PBCB_A_PP_NEXT];
4948 5181 2      PBCB [PBCB_A_PP_PREV] = PBCB [PBCB_A_PP_NEXT];
4949 5182 2
4950 5183 2      +
4951 5184 2      Initialize mode settings to default.
4952 5185 2      -
4953 5186 2      PBCB [PBCB_L_MODE_SETTINGS] = PBCB_K_DEF_MODE_SETTINGS;
4954 5187 2
4955 5188 2      +
4956 5189 2      Return the address of the PBCB we've built.
4957 5190 2      -
4958 5191 2      .PBCB_ADDR = .PBCB;
4959 5192 2
4960 5193 2      RETURN (SS$NORMAL);
4961 5194 1      END;

```

! Routine SMG\$CREATE\_PASTEBOARD

				00FC 00000	.ENTRY	SMG\$CREATE_PASTEBOARD, Save R2,R3,R4,R5,-	5063
						R6,R7	
				57 00000000G 00 9E 00002	MOVAB	LIB\$GET_VM, R7	
				5E 08 C2 00009	SUBL2	#8, SP	
				04 AE 04 9F 0000C	PUSHAB	PBCB	5129
				04 8F 3C 0000F	MOVZWL	#332, 4(SP)	
				04 AE 9F 00015	PUSHAB	4(SP)	
				67 02 FB 00018	CALLS	#2, LIB\$GET_VM	
				56 50 D0 0001B	MOVL	R0, STATUS	
				51 56 E9 0001E	BLBC	STATUS, 2\$	
014C 8F	00			6E 00 2C 00021	MOVCS	#0, (SP), #0, #332, @PBCB	5133
				04 BE 00028			
				04 08 C1 0002A	ADDL3	#8, PBCB, -(SP)	5139
				7E 04 AC 7D 0002F	MOVQ	ROWS, -(SP)	
				0000V CF 03 FB 00033	CALLS	#3, SMG\$CREATE_WCB	
				56 50 D0 00038	MOVL	R0, STATUS	
				21 56 E9 0003B	BLBC	STATUS, 1\$	
				52 04 AE D0 0003E	MOVL	PBCB, R2	5161
				6C A2 9F 00042	PUSHAB	108(R2)	
				04 AE 0100 8F 3C 00045	MOVZWL	#256, 4(SP)	5160
				04 AE 9F 0004B	PUSHAB	4(SP)	
				67 02 FB 0004E	CALLS	#2, LIB\$GET_VM	5161
				56 50 D0 00051	MOVL	R0, STATUS	
				1F 56 E8 00054	BLBS	STATUS, 3\$	5162
				08 A2 DD 00057	PUSHL	8(R2)	5170
0000V CF				01 FB 0005A	CALLS	#1, SMG\$DEALLOCATE_WCB	
				04 AE 9F 0005F 1\$:	PUSHAB	PBCB	5171



04	AE	014C	8F	3C	00062	MOVZWL	#332, 4(SP)	:	
		04	AE	9F	00068	PUSHAB	4(SP)	:	
00000000G	00		02	FB	0006B	CALLS	#2, LIB\$FREE_VM	:	
	50		56	D0	00072	MOVL	STATUS, R0	:	5172
				04	00075	RET		:	
	50	04	AE	D0	00076	MOVL	PBCB, R0	:	5175
70	A0	0100	8F	B0	0007A	MOVW	#256, 112(R0)	:	
	60		50	D0	00080	MOVL	R0, (R0)	:	5180
04	A0		50	D0	00083	MOVL	R0, 4(R0)	:	5181
0C	A0		02	D0	00087	MOVL	#2, 12(R0)	:	5186
0C	BC		50	D0	0008B	MOVL	R0, @PBCB_ADDR	:	5191
	50		01	D0	0008F	MOVL	#1, R0	:	5193
			04	00092	RET			:	5194

; Routine Size: 147 bytes,

Routine Base: \_SMG\$CODE + 1B0A

; 4962

5195 1 !<BLF/PAGE>



```

4964 5196 1 %SBTTL 'SMG$CREATE_VIRTUAL_DISPLAY - Create Virtual Display'
4965 5197 1 GLOBAL ROUTINE SMG$CREATE_VIRTUAL_DISPLAY (
4966 5198 1     NUM_ROWS,      ! height
4967 5199 1     NUM_COLS,      ! width
4968 5200 1     NEW_DISPLAY_ID,
4969 5201 1     DISPLAY_ATTRIBUTES,
4970 5202 1     VIDEO_ATTRIBUTES,
4971 5203 1     CHAR_SET
4972 5204 1 ) =
4973 5205 1
4974 5206 1 ++
4975 5207 1 FUNCTIONAL DESCRIPTION:
4976 5208 1     This routine creates a new virtual display -- returning its
4977 5209 1     assigned display_id. Its initial contents are blanks with
4978 5210 1     video attributes set to those specified. The cursor
4979 5211 1     will be at row 1 column 1.
4980 5212 1     This is the inner-most create_virtual_display routine. It
4981 5213 1     assumes all of its parameters are present.
4982 5214 1
4983 5215 1 CALLING SEQUENCE:
4984 5216 1     ret_status.wlc.v = SMG$CREATE_VIRTUAL_DISPLAY (
4985 5217 1         NUM_ROWS.rl.r,      ! Height
4986 5218 1         NUM_COLS.rl.r,      ! Width
4987 5219 1         NEW_DISPLAY_ID.wl.r,
4988 5220 1         DISPLAY_ATTRIBUTES.rl.r,
4989 5221 1         VIDEO_ATTRIBUTES.rl.r,
4990 5222 1         CHAR_SET.rl.r)
4991 5223 1
4992 5224 1 FORMAL PARAMETERS:
4993 5225 1     NUM_ROWS.rl.r    Number of rows in new virtual display.
4994 5226 1     NUM_COLS.rl.r    Number of columns in new virtual display.
4995 5227 1     NEW_DISPLAY_ID.wl.r Virtual display id of newly-created
4996 5228 1                       virtual display.
4997 5229 1     DISPLAY_ATTRIBUTES.rl.r The default display attributes.
4998 5230 1
4999 5231 1     SMG$M_BORDER if virtual display is to be
5000 5232 1                   displayed with a border.
5001 5233 1
5002 5234 1     SMG$M_TRUNC_ICON if an icon should be displayed
5003 5235 1                       when text overflows the display
5004 5236 1                       bounds.
5005 5237 1
5006 5238 1     SMG$M_DISPLAY_CONTROLS if carriage controls (CR, LF,
5007 5239 1                       TFF, VT, HT) should be displayed instead
5008 5240 1                       of executed.
5009 5241 1
5010 5242 1     VIDEO_ATTRIBUTES.rl.r The default rendition code to be
5011 5243 1                       applied to all output to this display unless
5012 5244 1                       overridden on a particular output call.
5013 5245 1
5014 5246 1 Values:
5015 5247 1
5016 5248 1
5017 5249 1
5018 5250 1
5019 5251 1
5020 5252 1
  
```



```

5021      5253 1 | SMG$M_BLINK displays characters blinking.
5022      5254 1 |
5023      5255 1 | SMG$M_BOLD displays characters in
5024      5256 1 | higher-than-normal intensity.
5025      5257 1 |
5026      5258 1 | SMG$M_REVERSE displays characters in reverse
5027      5259 1 | video -- that is, using the
5028      5260 1 | opposite default rendition of
5029      5261 1 | the virtual display.
5030      5262 1 |
5031      5263 1 | SMG$M_UNDERLINE displays characters underlined.
5032      5264 1 |
5033      5265 1 | CHAR_SET.rb.r Specifies the default character set to be used
5034      5266 1 | for this display.
5035      5267 1 | Recognized values are:
5036      5268 1 | SMG$C_UNITED_KINGDOM
5037      5269 1 | SMG$C_ASCII (default)
5038      5270 1 | SMG$C_SPEC_GRAPHICS
5039      5271 1 | SMG$C_ALT_CHAR
5040      5272 1 | SMG$C_ALT_GRAPHICS
5041      5273 1 |
5042      5274 1 | IMPLICIT INPUTS:
5043      5275 1 | NONE
5044      5276 1 |
5045      5277 1 | IMPLICIT OUTPUTS:
5046      5278 1 | NONE
5047      5279 1 |
5048      5280 1 | COMPLETION STATUS:
5049      5281 1 |
5050      5282 1 | SSS_NORMAL Normal successful completion
5051      5283 1 | LIB$INSVIRMEM Insufficient virtual memory to allocate needed
5052      5284 1 | buffer.
5053      5285 1 | SMG$INVARG Unrecognized Video Attributes
5054      5286 1 | or Unrecognized Display Attributes
5055      5287 1 |
5056      5288 1 | SIDE EFFECTS:
5057      5289 1 |
5058      5290 1 | NONE
5059      5291 1 |
5060      5292 1 | --
5061      5293 1 | BEGIN
5062      5294 2 | LOCAL
5063      5295 2 |
5064      5296 2 | STATUS, ! Status of subroutine calls
5065      5297 2 | DCB : REF $DCB_DECL, ! Addr of display control block
5066      5298 2 | DESC : REF BLOCK [8,BYTE]; ! Pointer to dynamic descriptor in
5067      5299 2 | ! DCB for border label
5068      5300 2 |
5069      5301 2 | +
5070      5302 2 | Allocate space for DCB itself. Quit if we can't get it.
5071      5303 2 | -
5072      5304 2 | IF NOT (STATUS = LIB$GET_VM ( %REF (DCB_K_SIZE), DCB))
5073      5305 2 | THEN
5074      5306 2 | RETURN (.STATUS);
5075      5307 2 |
5076      5308 2 | CH$FILL (0, DCB_K_SIZE, .DCB); ! set all fields to default of 0
5077      5309 2 |

```



```

5078 5310 2  +
5079 5311 2  | Set up dimensions of display
5080 5312 2  -
5081 5313 2  |   DCB [DCB_W_ROW_START] = 1;
5082 5314 2  |   DCB [DCB_W_NO_ROWS]  = ..NUM_ROWS;
5083 5315 2  |   DCB [DCB_W_COL_START] = 1;
5084 5316 2  |   DCB [DCB_W_NO_COLS]   = ..NUM_COLS;
5085 5317 2  |   DCB [DCB_L_BUFSIZE]   = ..NUM_ROWS * ..NUM_COLS;
5086 5318 2  |
5087 5319 2  +
5088 5320 2  | Record default display attributes, default video attributes and
5089 5321 2  | default character set.
5090 5322 2  -
5091 5323 2  |   DCB [DCB_B_DEF_DISPLAY_ATTR] = ..DISPLAY_ATTRIBUTES;
5092 5324 2  |   DCB [DCB_B_DEF_VIDEO_ATTR]  = ..VIDEO_ATTRIBUTES;
5093 5325 2  |   DCB [DCB_B_DEF_CHAR_SET]    = ..CHAR_SET;
5094 5326 2  |
5095 5327 2  +
5096 5328 2  | Set up various fields in the DCB
5097 5329 2  -
5098 5330 2  |   DCB [DCB_L_DID] = .DCB;           ! Display id itself -- the
5099 5331 2  |                                   ! address of the DCB
5100 5332 2  |
5101 5333 2  |   DCB [DCB_W_CURSOR_ROW] = 1;       ! Cursor row and column to home
5102 5334 2  |   DCB [DCB_W_CURSOR_COL] = 1;
5103 5335 2  |
5104 5336 2  |   DCB [DCB_B_STRUCT_TYPE] = DCB_K_STRUCT_TYPE; ! Mark as being a DCB
5105 5337 2  |   DCB [DCB_W_DCB_LENGTH]  = DCB_K_SIZE;       ! Size of structure
5106 5338 2  |
5107 5339 2  |   DCB [DCB_W_TOP_OF_SCRREG] = .DCB [DCB_W_ROW_START];
5108 5340 2  |   DCB [DCB_W_BOTTOM_OF_SCRREG] = .DCB [DCB_W_NO_ROWS];
5109 5341 2  |                                   ! init scrolling region
5110 5342 2  +
5111 5343 2  | Allocate enough space for both the text buffer and the attribute
5112 5344 2  | buffer.
5113 5345 2  -
5114 5346 2  |   IF NOT (STATUS = LIB$GET_VM (XREF (.DCB [DCB_L_BUFSIZE] * 2),
5115 5347 2  |                               DCB [DCB_A_TEXT_BUF]))
5116 5348 2  |   THEN
5117 5349 2  |     BEGIN ! Ran out of space
5118 5350 2  |       +
5119 5351 2  |       | If we can't get enough space for the buffers, we might as well
5120 5352 2  |       | give back the DCB space itself.
5121 5353 2  |       -
5122 5354 2  |       | LIB$FREE_VM (XREF (DCB_K_SIZE), DCB); ! Return DCB space
5123 5355 2  |       | RETURN (.STATUS); ! Return the LIB$INSVIRMEM from the
5124 5356 2  |       | ! buffer allocation attempt
5125 5357 2  |       | END; ! Ran out of space
5126 5358 2  |
5127 5359 2  +
5128 5360 2  | Use upper half of space allocated as the attribute buffer.
5129 5361 2  -
5130 5362 2  |   DCB [DCB_A_ATTR_BUF] = .DCB [DCB_A_TEXT_BUF] + .DCB [DCB_L_BUFSIZE];
5131 5363 2  |
5132 5364 2  +
5133 5365 2  | Initialize text and attribute buffers.
5134 5366 2  -

```



```
5135 5367 2 CH$FILL (%C' ' .DCB [DCB_L_BUFSIZE], .DCB [DCB_A_TEXT_BUF]);
5136 5368 2 CH$FILL (.DCB [DCB_B_DEF_VIDEO_ATTR], .DCB [DCB_L_BUFSIZE],
5137 5369 2 .DCB [DCB_A_ATTR_BUF]);
5138 5370 2
5139 5371 2
5140 5372 2 + If we are dealing with a non-standard character set, allocate the
5141 5373 2 char_set buffer. If we can't, bail out, giving back all the space
5142 5374 2 allocated on this transaction.
5143 5375 2 -
5144 5376 2 IF .DCB [DCB_B_DEF_CHAR_SET] NEQ 0
5145 5377 2 THEN
5146 5378 2 BEGIN ! Will need char_set buffer
5147 5379 4 IF NOT (STATUS = LIB$GET_VM ( .DCB [DCB_L_BUFSIZE],
5148 5380 4 .DCB [DCB_A_CHAR_SET_BUF]))
5149 5381 3 THEN
5150 5382 4 BEGIN ! Bailout
5151 5383 4 +
5152 5384 4 | If we can't get space for buffer we need, give back the
5153 5385 4 | the text and attribute buffer, and DCB itself before
5154 5386 4 | quitting.
5155 5387 4 -
5156 5388 4 LIB$FREE_VM (%REF (2 * .DCB [DCB_L_BUFSIZE]),
5157 5389 4 .DCB [DCB_A_TEXT_BUF]);
5158 5390 4 LIB$FREE_VM (%REF (DCB_R_SIZE), DCB);
5159 5391 4 RETURN (.STATUS);
5160 5392 3 END; ! Bailout
5161 5393 2
5162 5394 2 CH$FILL (.DCB [DCB_B_DEF_CHAR_SET], .DCB [DCB_L_BUFSIZE],
5163 5395 2 .DCB [DCB_A_CHAR_SET_BUF]);
5164 5396 2
5165 5397 2 END; ! Will need char_set buffer
5166 5398 2
5167 5399 2
5168 5400 2 +
5169 5401 2 | Allocate and clear the line characteristics vector.
5170 5402 2 -
5171 5403 2 IF NOT (STATUS = LIB$GET_VM (%REF (.DCB [DCB_W_NO_ROWS] + 1),
5172 5404 2 .DCB [DCB_A_LINE_CHAR]))
5173 5405 2 THEN
5174 5406 2 BEGIN ! Error path
5175 5407 2 +
5176 5408 2 | Give back all space accumulated on this trans. before
5177 5409 2 | bailing out.
5178 5410 2 -
5179 5411 2 LIB$FREE_VM (%REF (2 * .DCB [DCB_L_BUFSIZE]),
5180 5412 2 .DCB [DCB_A_TEXT_BUF]);
5181 5413 2
5182 5414 2 IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
5183 5415 2 THEN
5184 5416 2 LIB$FREE_VM (.DCB [DCB_L_BUFSIZE], .DCB [DCB_A_CHAR_SET_BUF]);
5185 5417 2
5186 5418 2 LIB$FREE_VM (%REF (DCB_K_SIZE), DCB);
5187 5419 2
5188 5420 2 RETURN (.STATUS);
5189 5421 2 END; ! Error path
5190 5422 2
5191 5423 2 CH$FILL ( 0, .DCB [DCB_W_NO_ROWS] + 1, .DCB [DCB_A_LINE_CHAR]);
```



```

5192 5424 2
5193 5425 2
5194 5426 2 !+ Initialize pasting queue headers to point to self.
5195 5427 2 !-
5196 5428 2 DCB [DCB_A_PP_NEXT] = DCB [DCB_A_PP_NEXT];
5197 5429 2 DCB [DCB_A_PP_PREV] = DCB [DCB_A_PP_NEXT];
5198 5430 2
5199 5431 2 !+
5200 5432 2 !- Initialize border label descriptor to virgin dynamic string descriptor
5201 5433 2 !-
5202 5434 2 DESC = DCB [DCB_Q_LABEL_DESC];
5203 5435 2
5204 5436 2 DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
5205 5437 2 DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
5206 5438 2
5207 5439 2 !+
5208 5440 2 !- Return the new display id to caller
5209 5441 2 !-
5210 5442 2 .NEW_DISPLAY_ID = .DCB;
5211 5443 2
5212 5444 2 RETURN (SS$_NORMAL);
5213 5445 1 END;

```

03FC 00000

```
.ENTRY SMG$$CREATE_VIRTUAL_DISPLAY, Save R2,R3,R4,-; 5197
```

			59	00000000G	00	9E	00002	MOVAB	R5,R6,R7,R8,R9		
			58	00000000G	00	9E	00009	MOVAB	LIB\$FREE_VM, R9		
			5E		08	C2	00010	SUBL2	LIB\$GET_VM, R8		
				04	AE	9F	00013	PUSHAB	#8, SP		
		04	AE		70	8F	9A	MOVZBL	DCB		5304
				04	AE	9F	0001B	PUSHAB	#112, 4(SP)		
			68		02	FB	0001E	CALLS	4(SP)		
			57		50	D0	00021	CALLS	#2, LIB\$GET_VM		
			03		57	E8	00024	MOVL	R0, STATUS		
					00F5	31	00027	BLBS	STATUS, 1\$		
			56		04	AE	D0	BRW	7\$		
0070	8F		6E			00	2C	MOVL	DCB, R6		5308
						66		MOVCS	#0, (SP), #0, #112, (R6)		
						01	B0				
		02	A6		04	BC	B0	MOVW	#1, (R6)		5313
		04	A6			01	B0	MOVW	@NUM_ROWS, 2(R6)		5314
		06	A6		08	BC	B0	MOVW	#1, 4(R6)		5315
		04	BC		08	BC	C5	MOVW	@NUM_COLS, 6(R6)		5316
	3C	A6	2F		10	BC	90	MULL3	@NUM_COLS, @NUM_ROWS, 60(R6)		5317
			2E		14	BC	90	MOVB	@DISPLAY_ATTRIBUTES, 47(R6)		5323
			30		18	BC	90	MOVB	@VIDEO_ATTRIBUTES, 46(R6)		5324
			38			56	D0	MOVB	@CHAR SET, 48(R6)		5325
			28			8F	D0	MOVL	R6, 56(R6)		5330
			44		00010001	11	90	MOVL	#65537, 40(R6)		5333
			45			8F	9B	MOVB	#17, 68(R6)		5336
			48		70	66	D0	MOVZBW	#112, 69(R6)		5337
					10	A6	9F	MOVL	(R6), 72(R6)		5339
						01	78	PUSHAB	16(R6)		5347
		04	AE	3C	A6			ASHL	#1, 60(R6), 4(SP)		5346



				04	AE	9F	0007F	PUSHAB	4(SP)			
			68		02	FB	00082	CALLS	#2, LIB\$GET_VM		5347	
			57		50	D0	00085	MOVL	R0, STATUS			
			03		57	E8	00088	BLBS	STATUS, 2\$			
					0083	31	0008B	BRW	6\$			
			56	04	AE	D0	0008E	MOVL	DCB, R6		5362	
3C	A6	14	A6	3C	A6	C1	00092	ADDL3	60(R6), 16(R6), 20(R6)			
			20		00	2C	00099	MOVC5	#0, (SP), #32, 60(R6), @16(R6)		5367	
3C	A6	2E	A6	10	B6		0009F					
			6E		00	2C	000A1	MOVC5	#0, (SP), 46(R6), 60(R6), @20(R6)		5369	
				14	B6		000A8					
				30	A6	95	000AA	TSTB	48(R6)		5376	
					2A	13	000AD	BEQL	4\$			
				18	A6	9F	000AF	PUSHAB	24(R6)		5380	
			68	3C	A6	9F	000B2	PUSHAB	60(R6)		5379	
			57		02	FB	000B5	CALLS	#2, LIB\$GET_VM		5380	
			0E		50	D0	000B8	MOVL	R0, STATUS			
					57	E8	000BB	BLBS	STATUS, 3\$			
				10	A6	9F	000BE	PUSHAB	16(R6)		5389	
		04	AE	3C	01	78	000C1	ASHL	#1, 60(R6), 4(SP)		5388	
				04	AE	9F	000C7	PUSHAB	4(SP)			
					42	11	000CA	BRB	5\$		5389	
			50	04	AE	D0	000CC	MOVL	DCB, R0		5394	
3C	A0	30	A0		00	2C	000D0	MOVC5	#0, (SP), 48(R0), 60(R0), @24(R0)		5395	
			6E		B0		000D7					
			52		04	AE	D0	000D9	MOVL	DCB, R2	5404	
					4C	A2	9F	000DD	PUSHAB	76(R2)		
				04	AE	02	A2	3C	000E0	MOVZWL	2(R2), 4(SP)	5403
					04	AE	D6	000E5	INCL	4(SP)		
				04	AE	9F	000E8	PUSHAB	4(SP)			
			68		02	FB	000EB	CALLS	#2, LIB\$GET_VM		5404	
			57		50	D0	000EE	MOVL	R0, STATUS			
			2F		57	E8	000F1	BLBS	STATUS, 8\$			
				10	A2	9F	000F4	PUSHAB	16(R2)		5412	
		04	AE	3C	01	78	000F7	ASHL	#1, 60(R2), 4(SP)		5411	
				04	AE	9F	000FD	PUSHAB	4(SP)			
			69		02	FB	00100	CALLS	#2, LIB\$FREE_VM		5412	
				18	A2	D5	00103	TSTL	24(R2)		5414	
					09	13	00106	BEQL	6\$			
				18	A2	9F	00108	PUSHAB	24(R2)		5416	
				3C	A2	9F	0010B	PUSHAB	60(R2)			
			69		02	FB	0010E	CALLS	#2, LIB\$FREE_VM		5418	
				04	AE	9F	00111	PUSHAB	DCB			
		04	AE	70	8F	9A	00114	MOVZBL	#112, 4(SP)			
				04	AE	9F	00119	PUSHAB	4(SP)			
			69		02	FB	0011C	CALLS	#2, LIB\$FREE_VM		5420	
			50		57	D0	0011F	MOVL	STATUS, R0			
					04		00122	RET				
			56	04	AE	D0	00123	MOVL	DCB, R6		5423	
			50	02	A6	3C	00127	MOVZWL	2(R6), R0			
					50	D6	0012B	INCL	R0			
50			6E		00	2C	0012D	MOVC5	#0, (SP), #0, R0, @76(R6)			
				4C	B6		00132					
			20	20	A6	9E	00134	MOVAB	32(R6), 32(R6)		5428	
			24	20	A6	9E	00139	MOVAB	32(R6), 36(R6)		5429	
			50	08	A6	9E	0013E	MOVAB	8(R6), DESC		5434	
		02	A0	020E	8F	B0	00142	MOVW	#526, 2(DESC)		5437	



SMG\$DISPLAY\_LIN SMG\$DISPLAY\_LINKS - Virtual Display Linkages 16-Sep-1984 00:29:22 VAX-11 Bliss-32 V4.0-742 Page 150  
1-096 SMG\$\$CREATE\_VIRTUAL\_DISPLAY - Create Virtual Di 14-Sep-1984 13:09:43 [SMGRTL.SRC]SMGDISLIN.B32;1 (30)

0C	BC	56	D0	00148	MOVL	R6, @NEW_DISPLAY_ID	:	5442
	50	01	D0	0014C	MOVL	#1, R0	:	5444
		04	0014F	RET			:	5445

; Routine Size: 336 bytes, Routine Base: \_SMG\$CODE + 1B9D

; 5214 5446 1 !<BLF/PAGE>



SMG\$DISPLAY_LIN	SMG\$DISPLAY_LINKS - Virtual Display Linkages	F 7	16-Sep-1984 00:29:22	VAX-11 Bliss-32 V4.0-742	Page 151
1-096	SMG\$\$CREATE_WCB - Create WCB and its buffers		14-Sep-1984 13:09:43	[SMGRTL.SRC]SMGDISLIN.B32;1	(31)

```

5216 5447 1 %SBTTL 'SMG$$CREATE_WCB - Create WCB and its buffers'
5217 5448 1 GLOBAL ROUTINE SMG$$CREATE_WCB (
5218 5449 1     ROWS,
5219 5450 1     COLS,
5220 5451 1     WCB_ADDR
5221 5452 1 ) =
5222 5453 1 ++
5223 5454 1 FUNCTIONAL DESCRIPTION:
5224 5455 1
5225 5456 1     This routine allocates space for the window control block and
5226 5457 1     its window text and attribute buffers and initializes them.
5227 5458 1     Two sets of these two buffers are built -- one to reflect what
5228 5459 1     is currently on the screen and one to build up what the next
5229 5460 1     screen image should look like.
5230 5461 1
5231 5462 1 CALLING SEQUENCE:
5232 5463 1
5233 5464 1     ret_status.wlc.v = SMG$$CREATE_WCB (     ROWS.rl.r,
5234 5465 1                                           COLS.rl.r,
5235 5466 1                                           WCB_ADDR.wl.r)
5236 5467 1
5237 5468 1 FORMAL PARAMETERS:
5238 5469 1
5239 5470 1     ROWS.rl.r      No. of rows in each of the buffers
5240 5471 1
5241 5472 1     COLS.rl.r      No. of columns in each of the buffers
5242 5473 1
5243 5474 1     WCB_ADDR.wl.r  Address of the newly-created WCB -- returned to
5244 5475 1     caller.
5245 5476 1
5246 5477 1
5247 5478 1 IMPLICIT INPUTS:
5248 5479 1
5249 5480 1     NONE
5250 5481 1
5251 5482 1 IMPLICIT OUTPUTS:
5252 5483 1
5253 5484 1     NONE
5254 5485 1
5255 5486 1 COMPLETION STATUS:
5256 5487 1
5257 5488 1
5258 5489 1     $$$ NORMAL      Normal successful completion
5259 5490 1     LIB$_INSVIRMEM  Insufficient virtual memory to allocate needed
5260 5491 1                     buffer.
5261 5492 1
5262 5493 1 SIDE EFFECTS:
5263 5494 1
5264 5495 1     NONE
5265 5496 1
5266 5497 2 --
5267 5498 2 BEGIN
5268 5499 2 LOCAL
5269 5500 2     WCB : REF $WCB_DECL,      ! Address of WCB allocated.
5270 5501 2     STATUS;                  ! Status of subroutine calls
5271 5502 2
5272 5503 2 !+ Allocate space for the WCB itself.

```



```

5273 5504 2  !-
5274 5505 3  IF NOT (STATUS = LIB$GET_VM (%REF (WCB_K_SIZE), WCB))
5275 5506 4  THEN
5276 5507 5  RETURN (.STATUS);
5277 5508 6
5278 5509 7  CH$FILL (0, WCB_K_SIZE, .WCB);      ! Clear all fields to default 0
5279 5510 8
5280 5511 9  WCB [WCB_L_BUFSIZE] = ..ROWS * ..COLS; ! Overall size of each buffer
5281 5512 10
5282 5513 11  !+
5283 5514 12  Attempt to get space for all 4 buffers at once, returning an error if
5284 5515 13  we can't.
5285 5516 14  !-
5286 5517 15  IF NOT (STATUS = LIB$GET_VM ( %REF (4 * .WCB [WCB_L_BUFSIZE]),
5287 5518 16  WCB [WCB_A_TEXT_BUF]))
5288 5519 17  THEN
5289 5520 18  BEGIN      ! No more space
5290 5521 19  !+
5291 5522 20  If we can't get space for buffers, we might as well give back
5292 5523 21  the WCB space itself.
5293 5524 22  !-
5294 5525 23  LIB$FREE_VM (%REF (WCB_K_SIZE), WCB);
5295 5526 24  RETURN (.STATUS);
5296 5527 25  END;      ! No more space
5297 5528 26
5298 5529 27  !+
5299 5530 28  Carve up the space gotten into the 4 buffers we need.
5300 5531 29  !-
5301 5532 30  WCB [WCB_A_ATTR_BUF] = .WCB [WCB_A_TEXT_BUF] + .WCB [WCB_L_BUFSIZE];
5302 5533 31  WCB [WCB_A_SCR_TEXT_BUF] = .WCB [WCB_A_TEXT_BUF] + 2 * .WCB [WCB_L_BUFSIZE];
5303 5534 32  WCB [WCB_A_SCR_ATTR_BUF] = .WCB [WCB_A_TEXT_BUF] + 3 * .WCB [WCB_L_BUFSIZE];
5304 5535 33
5305 5536 34  !+
5306 5537 35  Initialize the working buffers.
5307 5538 36  !-
5308 5539 37  CH$FILL (%C' ', .WCB [WCB_L_BUFSIZE], .WCB [WCB_A_TEXT_BUF]);
5309 5540 38  CH$FILL (0, .WCB [WCB_L_BUFSIZE], .WCB [WCB_A_ATTR_BUF]);
5310 5541 39
5311 5542 40  !+
5312 5543 41  Initialize the buffers representing what's on the screen to non-
5313 5544 42  matchable text as an initial state. This means the first time
5314 5545 43  minimum screen update looks at it it will cause the entire window
5315 5546 44  to be repainted.
5316 5547 45  !-
5317 5548 46  CH$FILL (-1, .WCB [WCB_L_BUFSIZE], .WCB [WCB_A_SCR_TEXT_BUF]);
5318 5549 47  CH$FILL (0, .WCB [WCB_L_BUFSIZE], .WCB [WCB_A_SCR_ATTR_BUF]);
5319 5550 48
5320 5551 49  !+
5321 5552 50  Allocate the line characteristic vectors. There are two of them --
5322 5553 51  one for the text buffer and one for the screen text buffer. We
5323 5554 52  allocate and initialize them together for efficiency.
5324 5555 53  !-
5325 5556 54  IF NOT (STATUS = LIB$GET_VM ( %REF (2 * (.ROWS + 1)),
5326 5557 55  WCB [WCB_A_LINE_CHAR]))
5327 5558 56  THEN
5328 5559 57  BEGIN      ! Error path
5329 5560 58  !+

```



```

5330 5561 3      ! Give back all space accumulated on this transaction.
5331 5562      !
5332 5563      LIB$FREE_VM ( %REF (4 * .WCB [WCB_L_BUFSIZE]),
5333 5564                  WCB [WCB_A_TEXT_BUF]);
5334 5565
5335 5566      LIB$FREE_VM ( %REF (WCB_K_SIZE), WCB);
5336 5567      END;      ! Error path
5337 5568
5338 5569      + Clear both buffer to zero at once
5339 5570      -
5340 5571      CH$FILL (0, 2 * (..ROWS + 1), .WCB [WCB_A_LINE_CHAR]);
5341 5572
5342 5573      +
5343 5574      Use upper half of space just allocated and cleared as the line
5344 5575      characteristics vector for the screen text buffer.
5345 5576      -
5346 5577      WCB [WCB_A_SCR_LINE_CHAR] = .WCB [WCB_A_LINE_CHAR] + ..ROWS + 1;
5347 5578
5348 5579      +
5349 5580      Fill in rest of WCB and return the address of the WCB we've built.
5350 5581      -
5351 5582      WCB [WCB_W_NO_ROWS] = ..ROWS;
5352 5583      WCB [WCB_W_ROW_START] = 1;
5353 5584      WCB [WCB_W_NO_COLS] = ..COLS;
5354 5585      WCB [WCB_W_COL_START] = 1;
5355 5586
5356 5587      .WCB_ADDR = .WCB;
5357 5588
5358 5589      RETURN (SS$_NORMAL);
5359 5590      END;

```

! Routine SMG\$CREATE\_WCB

				OFFC 00000	.ENTRY	SMG\$CREATE_WCB, Save R2,R3,R4,R5,R6,R7,R8,-;	5448
		5B	00000000G	00 9E 00002	MOVAB	R9,R10,R11	
		5E		08 C2 00009	SUBL2	LIB\$FREE_VM, R11	
			04	AE 9F 0000C	PUSHAB	#8, SP	5505
	04	AE		34 D0 0000F	MOVL	WCB	
			04	AE 9F 00013	PUSHAB	#52, 4(SP)	
				02 FB 00016	CALLS	4(SP)	
	00000000G	00		50 D0 0001D	CALLS	#2, LIB\$GET_VM	
		5A		5A E9 00020	MOVL	R0, STATUS	
		3A		AE D0 00023	BLBC	STATUS, 1\$	
		56	04	00 2C 00027	MOVL	WCB, R6	5509
34	00	6E		66 0002C	MOVCS	#0, (SP), #0, #52, (R6)	
		59	04	BC D0 0002D	MOVL	@ROWS, R9	5511
	28	A6		08 BC C5 00031	MULL3	@COLS, R9, 40(R6)	
			08	A6 9F 00037	PUSHAB	8(R6)	5518
	04	AE	28	02 78 0003A	ASHL	#2, 40(R6), 4(SP)	5517
			04	AE 9F 00040	PUSHAB	4(SP)	
	00000000G	00		02 FB 00043	CALLS	#2, LIB\$GET_VM	5518
		5A		50 D0 0004A	MOVL	R0, STATUS	
		11		5A E8 0004D	BLBS	STATUS, 2\$	
			04	AE 9F 00050	PUSHAB	WCB	5525



		04	AE		34	D0	00053	MOVL	#52, 4(SP)		
				04	AE	9F	00057	PUSHAB	4(SP)		
		68			02	FB	0005A	CALLS	#2, LIB\$FREE_VM		
		50			5A	D0	0005D	MOVL	STATUS, R0	5526	
						04	00060	RET			
		56		04	AE	D0	00061	MOVL	WCB, R6	5532	
		57		08	A6	9E	00065	MOVAB	8(R6), R7		
		58		28	A6	9E	00069	MOVAB	40(R6), R8		
	0C	A6			68	C1	0006D	ADDL3	(R8), (R7), 12(R6)		
		67			68	D0	00072	MOVL	(R8), R0	5533	
		50		14	A6	3E	00075	MOVAB	@0(R7)[R0], 20(R6)		
		68			03	C5	0007B	MULL3	#3, (R8), R0	5534	
	18	A6			67	C1	0007F	ADDL3	(R7), R0, 24(R6)		
68		20			00	2C	00084	MOVC5	#0, (SP), #32, (R8), @0(R7)	5539	
		6E			00	B7	00089				
68		00			00	2C	0008B	MOVC5	#0, (SP), #0, (R8), @12(R6)	5540	
		6E			0C	B6	00090				
68	FF	8F			00	2C	00092	MOVC5	#0, (SP), #-1, (R8), @20(R6)	5548	
		6E			14	B6	00098				
68		00			00	2C	0009A	MOVC5	#0, (SP), #0, (R8), @24(R6)	5549	
		6E			18	B6	0009F				
		52			2C	A6	9F	000A1	PUSHAB	44(R6)	5557
						01	78	000A4	ASHL	#1, R9, R2	5556
		52			02	C0	000A8	ADDL2	#2, R2		
				04	AE	D0	000AB	MOVL	R2, 4(SP)		
					04	AE	9F	000AF	PUSHAB	4(SP)	
		00000000G			02	FB	000B2	CALLS	#2, LIB\$GET_VM	5557	
					50	D0	000B9	MOVL	R0, STATUS		
		1A			5A	E8	000BC	BLBS	STATUS, 3\$		
					57	DD	000BF	PUSHL	R7	5564	
	04	AE			02	78	000C1	ASHL	#2, (R8), 4(SP)	5563	
					04	AE	9F	000C6	PUSHAB	4(SP)	
		68			02	FB	000C9	CALLS	#2, LIB\$FREE_VM	5564	
					04	AE	9F	000CC	PUSHAB	WCB	5566
				04	AE	D0	000CF	MOVL	#52, 4(SP)		
		68			04	AE	9F	000D3	PUSHAB	4(SP)	
		57			02	FB	000D6	CALLS	#2, LIB\$FREE_VM		
52		6E			04	AE	D0	000D9	MOVL	WCB, R7	5571
					00	2C	000DD	MOVC5	#0, (SP), #0, R2, @44(R7)		
		50			2C	B7	000E2				
					2C	A7	C1	000E4	ADDL3	44(R7), R9, R0	5577
		30			01	A0	9E	000E9	MOVAB	1(R0), 48(R7)	
		02			59	B0	000EE	MOVW	R9, 2(R7)	5582	
		67			01	B0	000F2	MOVW	#1, (R7)	5583	
					08	BC	B0	000F5	MOVW	@COLS, 6(R7)	5584
		06			01	B0	000FA	MOVW	#1, 4(R7)	5585	
		04			57	D0	000FE	MOVL	R7, @WCB_ADDR	5587	
		0C			01	D0	00102	MOVL	#1, R0	5589	
		50			04	00	00105	RET		5590	

; Routine Size: 262 bytes, Routine Base: \_SMG\$CODE + 1CED

; 5360 5591 1 !<BLF/PAGE>



```

5362 5592 1 %SBTTL 'SMG$$DEALLOCATE_WCB - Get rid of WCB and its buffers'
5363 5593 1 GLOBAL ROUTINE SMG$$DEALLOCATE_WCB (WCB : REF $WCB_DECL) =
5364 5594 1 ++
5365 5595 1 FUNCTIONAL DESCRIPTION:
5366 5596 1
5367 5597 1 This routine deallocates space for the window control block and
5368 5598 1 its window text and attribute buffers.
5369 5599 1
5370 5600 1 CALLING SEQUENCE:
5371 5601 1
5372 5602 1 ret_status.wlc.v = SMG$$CREATE_WCB ( WCB.wl.r)
5373 5603 1
5374 5604 1 FORMAL PARAMETERS:
5375 5605 1
5376 5606 1 WCB.wl.r Address of the previously-created WCB.
5377 5607 1
5378 5608 1 IMPLICIT INPUTS:
5379 5609 1
5380 5610 1 contents of WCB
5381 5611 1
5382 5612 1 IMPLICIT OUTPUTS:
5383 5613 1
5384 5614 1 NONE
5385 5615 1
5386 5616 1 COMPLETION STATUS:
5387 5617 1
5388 5618 1 $$$ NORMAL Normal successful completion
5389 5619 1 LIB$_xxx Errors from LIB$FREE_VM
5390 5620 1
5391 5621 1 SIDE EFFECTS:
5392 5622 1
5393 5623 1 NONE
5394 5624 1 --
5395 5625 1 BEGIN
5396 5626 1 LOCAL
5397 5627 1 RET_STATUS, ! Status to be returned to caller
5398 5628 1 STATUS; ! Status of subroutine calls
5399 5629 1
5400 5630 1 +
5401 5631 1 Attempt to deallocate the space for all 4 buffers (text and attr) at
5402 5632 1 once.
5403 5633 1 -
5404 5634 1 RET_STATUS = LIB$FREE_VM ( %REF(4 * .WCB [WCB_L_BUFSIZE]),
5405 5635 1 WCB [WCB_A_TEXT_BUF]);
5406 5636 1
5407 5637 1 +
5408 5638 1 Attempt to deallocate the alternate character set buffers if they
5409 5639 1 exist.
5410 5640 1 -
5411 5641 1 IF .WCB [WCB_A_CHAR_SET_BUF] NEQ 0
5412 5642 1 THEN
5413 5643 1 BEGIN ! Free alt char set buffers
5414 5644 1 ! NOTE: Right now we free them separately. If it turns out
5415 5645 1 ! they are allocated as a adjacent pair, we can deallocate
5416 5646 1 ! them with a single call.
5417 5647 1
5418 5648 1 STATUS = LIB$FREE_VM ( WCB [WCB_L_BUFSIZE],

```



```

5419      5649      3      WCB [WCB_A_CHAR_SET_BUF]);
5420      5650
5421      5651      IF NOT .STATUS THEN RET_STATUS = .STATUS ; ! Propagate an error
5422      5652      status
5423      5653
5424      5654      STATUS = LIB$FREE_VM ( WCB [WCB_L_BUFSIZE],
5425      5655      WCB [WCB_A_SCR_CHAR_SET_BUF]);
5426      5656
5427      5657      IF NOT .STATUS THEN RET_STATUS = .STATUS ; ! Propagate an error
5428      5658      status
5429      5659      END; ! Free alt char set buffers
5430      5660
5431      5661      !+
5432      5662      Deallocate the line characteristics vectors. These were allocated
5433      5663      as a pair so can be deallocated as a pair.
5434      5664      !-
5435      5665      STATUS = LIB$FREE_VM ( %REF ( 2 * (.WCB [WCB_W_NO_ROWS] + 1)),
5436      5666      WCB [WCB_A_LINE_CHAR]);
5437      5667
5438      5668      IF NOT .STATUS THEN RET_STATUS = .STATUS; ! Propagate an error
5439      5669      status
5440      5670
5441      5671      !+
5442      5672      Deallocate the WCB itself.
5443      5673      !-
5444      5674      STATUS = LIB$FREE_VM (%REF(WCB_K_SIZE), WCB);
5445      5675      IF NOT .STATUS THEN RET_STATUS = .STATUS ; ! Propagate an error status
5446      5676
5447      5677      RETURN (.RET_STATUS);
5448      5678
5449      5679      1      END; ! Routine SMG$$DEALLOCATE_WCB
  
```

				001C 00000	.ENTRY SMG\$\$DEALLOCATE_WCB, Save R2,R3,R4	5593
		54	00000000G	00 9E 00002	MOVAB LIB\$FREE_VM, R4	
		5E		04 C2 00009	SUBL2 #4, SP	
		52	04	AC D0 0000C	MOVL WCB, R2	5635
			08	A2 9F 00010	PUSHAB 8(R2)	
04	AE	28	A2	02 78 00013	ASHL #2, 40(R2), 4(SP)	5634
			04	AE 9F 00019	PUSHAB 4(SP)	
		64		02 FB 0001C	CALLS #2, LIB\$FREE_VM	5635
		53		50 D0 0001F	MOVL R0, RET_STATUS	
			10	A2 D5 00022	TSTL 16(R2)	5641
				1E 13 00025	BEQL 2\$	
			10	A2 9F 00027	PUSHAB 16(R2)	5649
			28	A2 9F 0002A	PUSHAB 40(R2)	5648
		64		02 FB 0002D	CALLS #2, LIB\$FREE_VM	5649
		03		50 E8 00030	BLBS STATUS, 1\$	5651
		53		50 D0 00033	MOVL STATUS, RET_STATUS	
			1C	A2 9F 00036 1\$:	PUSHAB 28(R2)	5655
			28	A2 9F 00039	PUSHAB 40(R2)	5654
		64		02 FB 0003C	CALLS #2, LIB\$FREE_VM	5655
		03		50 E8 0003F	BLBS STATUS, 2\$	5657
		53		50 D0 00042	MOVL STATUS, RET_STATUS	



```

SMG$DISPLAY_LIN SMG$DISPLAY_LINKS - Virtual Display Linkages 16-Sep-1984 00:29:22 VAX-11 Bliss-32 V4.0-742
1-096 SMG$DEALLOCATE_WCB - Get rid of WCB and its bu 14-Sep-1984 13:09:43 [SMGRTL.SRC]SMGDISLIN.B32;1

```

			2C	A2	9F	00045	2\$:	PUSHAB	44(R2)	:	5666
			02	A2	3C	00048		MOVZWL	2(R2), R1	:	5665
04	AE			01	78	0004C		ASHL	#1, R1, 4(SP)	:	
		04		02	C0	00051		ADDL2	#2, 4(\$P)	:	
			04	AE	9F	00055		PUSHAB	4(\$P)	:	
				02	FB	0005B		CALLS	#2, LIB\$FREE_VM	:	5666
				50	E8	0005B		BLBS	STATUS, 3\$	:	5668
				50	D0	0005E		MOVL	STATUS, RET_STATUS	:	
			04	AC	9F	00061	3\$:	PUSHAB	WCB	:	5674
				34	D0	00064		MOVL	#52, 4(SP)	:	
		04		AE	9F	00068		PUSHAB	4(\$P)	:	
				02	FB	0006B		CALLS	#2, LIB\$FREE_VM	:	
				50	E8	0006E		BLBS	STATUS, 4\$	:	5675
				50	D0	00071		MOVL	STATUS, RET_STATUS	:	
				53	D0	00074	4\$:	MOVL	RET_STATUS, -R0	:	5677
				04	00	0077		RET		:	5679

: 5450 5680 1 !&lt;BLF/PAGE&gt;



```

5452 5681 1 %SBTTL 'SMG$$DUPL_VIRTUAL_DISPLAY - Duplicate a virtual display'
5453 5682 1 GLOBAL ROUTINE SMG$$DUPL_VIRTUAL_DISPLAY (
5454 5683 1     CURR_DISPLAY_ID,
5455 5684 1     NEW_DISPLAY_ID
5456 5685 1 ) =
5457 5686 1
5458 5687 1 ++
5459 5688 1 FUNCTIONAL DESCRIPTION:
5460 5689 1     This routine makes a copy of an existing virtual display,
5461 5690 1     assigning it a new virtual display number. The new virtual
5462 5691 1     will not be pasted anywhere -- even if the virtual display from
5463 5692 1     which it was created was.
5464 5693 1
5465 5694 1 CALLING SEQUENCE:
5466 5695 1
5467 5696 1     ret_status.wlc.v = SMG$$DUPL_VIRTUAL_DISPLAY (CURR_DISPLAY_ID,
5468 5697 1     NEW_DISPLAY_ID)
5469 5698 1
5470 5699 1 FORMAL PARAMETERS:
5471 5700 1
5472 5701 1     CURR_DISPLAY_ID.rl.r    Display id of virtual display to be
5473 5702 1                          replicated.
5474 5703 1
5475 5704 1     NEW_DISPLAY_ID.wl.r    Display id of newly-created virtual
5476 5705 1                          display.
5477 5706 1
5478 5707 1 IMPLICIT INPUTS:
5479 5708 1
5480 5709 1     NONE
5481 5710 1
5482 5711 1 IMPLICIT OUTPUTS:
5483 5712 1
5484 5713 1     NONE
5485 5714 1
5486 5715 1 COMPLETION STATUS:
5487 5716 1
5488 5717 1     SSS NORMAL             Normal successful completion
5489 5718 1     LIB$INSVIRMEM          Insufficient virtual memory to allocate needed
5490 5719 1                          buffer.
5491 5720 1
5492 5721 1 SIDE EFFECTS:
5493 5722 1
5494 5723 1     NONE
5495 5724 1 --
5496 5725 2 BEGIN
5497 5726 2 LOCAL
5498 5727 2     DCB      : REF $DCB_DECL,      ! Address of current DCB.
5499 5728 2     DCB_NEW  : REF $DCB_DECL,      ! Address of new DCB
5500 5729 2     STATUS;   ! Status of subroutine calls
5501 5730 2
5502 5731 2     $SMG$GET_DCB (.CURR_DISPLAY_ID, DCB); ! Get addr of DCB for current
5503 5732 2                          ! display
5504 5733 2
5505 5734 2 ++
5506 5735 2     If a backup DCB does not yet exist, allocate one.
5507 5736 2     Make a new virtual display using the sizes and attributes of the old
5508 5737 2     one. Quit if we can't.
  
```



```

5509 5738 2  !-
5510 5739 2  IF .DCB [DCB_A_BACKUP_DCB] EQL 0
5511 5740 2  THEN
5512 5741 3  BEGIN ! 1st time, create the backup
5513 5742 4  IF NOT (STATUS = SMG$$CREATE_VIRTUAL_DISPLAY (
5514 5743 4  %REF (.DCB [DCB_W_NO_ROWS]),      ! #rows
5515 5744 4  %REF (.DCB [DCB_W_NO_COLS]),      ! #cols
5516 5745 4  .NEW_DISPLAY_ID,                  ! new id
5517 5746 4  %REF (.DCB [DCB_B_DEF_DISPLAY_ATTR]), ! disp
5518 5747 4  %REF (.DCB [DCB_B_DEF_VIDEO_ATTR]), ! video
5519 5748 4  %REF (.DCB [DCB_B_DEF_CHAR_SET])) ! alt char set
5520 5749 4  THEN
5521 5750 4  RETURN (.STATUS);
5522 5751 4
5523 5752 4  $SMG$GET_DCB (.NEW_DISPLAY_ID, DCB_NEW); ! Get DCB address of new
5524 5753 4
5525 5754 4  !+
5526 5755 4  ! Store the new display id in the new DCB.
5527 5756 4  !-
5528 5757 4  DCB_NEW [DCB_L_DID] = ..NEW_DISPLAY_ID;
5529 5758 4
5530 5759 4  END ! 1st time, create the backup
5531 5760 4
5532 5761 4  ELSE
5533 5762 4  BEGIN ! Backup already exists
5534 5763 4  .NEW_DISPLAY_ID = .DCB [DCB_A_BACKUP_DCB]; ! Return id of existing
5535 5764 4  DCB_NEW = .DCB [DCB_A_BACKUP_DCB];
5536 5765 4  END; ! Backup already exists
5537 5766 4
5538 5767 4  !+
5539 5768 4  ! Now need to copy over the current text and attribute buffers from
5540 5769 4  ! the current to the new.
5541 5770 4  !-
5542 5771 4
5543 5772 4  CH$MOVE ( .DCB [DCB_L_BUFSIZE],      ! #bytes
5544 5773 4  .DCB [DCB_A_TEXT_BUF],              ! from
5545 5774 4  .DCB_NEW [DCB_A_TEXT_BUF]);         ! to
5546 5775 4
5547 5776 4  CH$MOVE ( .DCB [DCB_L_BUFSIZE],      ! #bytes
5548 5777 4  .DCB [DCB_A_ATTR_BUF],              ! from
5549 5778 4  .DCB_NEW [DCB_A_ATTR_BUF]);         ! to
5550 5779 4
5551 5780 4  !+
5552 5781 4  ! Copy over the line characteristics vector.
5553 5782 4  !-
5554 5783 4  CH$MOVE ( .DCB [DCB_W_NO_ROWS] + 1,
5555 5784 4  .DCB [DCB_A_LINE_CHAR],
5556 5785 4  .DCB_NEW [DCB_A_LINE_CHAR]);
5557 5786 4
5558 5787 4  !+
5559 5788 4  ! Copy over stuff relating to borders and labels.
5560 5789 4  !-
5561 5790 4  IF .DCB_NEW [DCB_V_BORDERED]
5562 5791 4  THEN
5563 5792 4  BEGIN ! Bordered
5564 5793 4  LOCAL
5565 5794 4  DESC : REF BLOCK [8,BYTE]; ! Pointer to dynamic string

```



```

: 5566      5795 3      : descriptor in DCB for border
: 5567      5796 3      : label
: 5568      5797 3
: 5569      5798 3      DESC = DCB [DCB_Q_LABEL_DESC];
: 5570      5799 3
: 5571      5800 3      !+
: 5572      5801 3      ! If label exists, make a copy.
: 5573      5802 3      !-
: 5574      5803 3      IF .DESC [DSC$A_POINTER] NEQ 0
: 5575      5804 3      THEN
: 5576      5805 4          BEGIN      ! Labeled
: 5577      5806 5              IF NOT (STATUS = LIB$SCOPY_DXDX ( .DESC,
: 5578      5807 5                  DCB_NEW [DCB_Q_LABEL_DESC] ))
: 5579      5808 4              THEN
: 5580      5809 4                  RETURN (.STATUS);
: 5581      5810 4
: 5582      5811 4              DCB_NEW [DCB_W_LABEL_UNITS] = .DCB [DCB_W_LABEL_UNITS];
: 5583      5812 4              DCB_NEW [DCB_B_LABEL_POS] = .DCB [DCB_B_LABEL_POS];
: 5584      5813 4              DCB_NEW [DCB_B_LABEL_CHAR_SET] = .DCB [DCB_B_LABEL_CHAR_SET];
: 5585      5814 4              DCB_NEW [DCB_V_LABEL_CENTER] = .DCB [DCB_V_LABEL_CENTER];
: 5586      5815 4
: 5587      5816 3          END;      ! Labeled
: 5588      5817 2      END;      ! Bordered
: 5589      5818 2
: 5590      5819 2      !+
: 5591      5820 2      ! If alternate character set buffer involved, copy it over as well.
: 5592      5821 2      !-
: 5593      5822 2      IF .DCB_NEW [DCB_A_CHAR_SET_BUF] NEQ 0
: 5594      5823 2      THEN
: 5595      5824 3          BEGIN      ! Alt char set buffer involved
: 5596      5825 4              IF NOT (STATUS = LIB$GET_VM (DCB [DCB_L_BUFSIZE],
: 5597      5826 4                  DCB_NEW [DCB_A_CHAR_SET_BUF]))
: 5598      5827 3              THEN
: 5599      5828 3                  RETURN (.STATUS);
: 5600      5829 3
: 5601      5830 3          CH$MOVE (.DCB [DCB_L_BUFSIZE],      ! Num.
: 5602      5831 3              .DCB [DCB_A_CHAR_SET_BUF],      ! From
: 5603      5832 3              .DCB_NEW [DCB_A_CHAR_SET_BUF]);    ! To
: 5604      5833 3
: 5605      5834 2      END;      ! Alt char set buffer involved
: 5606      5835 2
: 5607      5836 2      !+
: 5608      5837 2      ! Also preserve the current cursor position.
: 5609      5838 2      !-
: 5610      5839 2      DCB_NEW [DCB_W_CURSOR_ROW] = .DCB [DCB_W_CURSOR_ROW];
: 5611      5840 2      DCB_NEW [DCB_W_CURSOR_COL] = .DCB [DCB_W_CURSOR_COL];
: 5612      5841 2
: 5613      5842 2      RETURN (SS$_NORMAL);
: 5614      5843 1      END;      ! Routine SMG$$DUPL_VIRTUAL_DISPLAY
```

```

                                03FC 00000
59 00000000G 8F D0 00002
```

```

.ENTRY SMG$$DUPL_VIRTUAL_DISPLAY, Save R2,R3,R4,- ; 5682
R5,R6,R7,R8,R9
MOVL #SMG$_INVDIS_ID, R9 ;
```



SMG\$DISPLAY\_LIN  
1-096

SMG\$DISPLAY\_LINKS - Virtual Display Linkages  
SMG\$\$DUPL\_VIRTUAL\_DISPLAY - Duplicate a virtual

C 8  
16-Sep-1984 00:29:22  
14-Sep-1984 13:09:43

VAX-11 Bliss-32 V4.0-742  
[SMGRTL.SRC]SMGDISLIN.B32;1

Page 161  
(33)

		5E		14	C2	00009	SUBL2	#20, SP		
		50		BC	D0	0000C	MOVL	@CURR_DISPLAY_ID, R0	5731	
	04	BC		38	A0	D1 00010	CMPL	56(R0), @CURR_DISPLAY_ID		
					06	12 00015	BNEQ	1\$		
		11		44	A0	91 00017	CMPB	68(R0), #17		
					04	13 0001B	BEQL	2\$		
		50		59	D0	0001D	MOVL	R9, R0		
					04	00020	RET			
		56		04	BC	D0 00021	MOVL	@CURR_DISPLAY_ID, DCB		
		50		40	A6	D0 00025	MOVL	64(DCB), R0	5739	
					56	12 00029	BNEQ	5\$		
	10	AE		30	A6	9A 0002B	MOVZBL	48(DCB), 16(SP)	5748	
				10	AE	9F 00030	PUSHAB	16(SP)		
	10	AE		2E	A6	9A 00033	MOVZBL	46(DCB), 16(SP)	5747	
				10	AE	9F 00038	PUSHAB	16(SP)		
	10	AE		2F	A6	9A 0003B	MOVZBL	47(DCB), 16(SP)	5746	
				10	AE	9F 00040	PUSHAB	16(SP)		
				08	AC	DD 00043	PUSHL	NEW_DISPLAY_ID	5745	
	14	AE		06	A6	3C 00046	MOVZWL	6(DCB), 20(SP)	5744	
				14	AE	9F 0004B	PUSHAB	20(SP)		
	14	AE		02	A6	3C 0004E	MOVZWL	2(DCB), 20(SP)	5743	
				14	AE	9F 00053	PUSHAB	20(SP)		
FCD7		CF		06	FB	00056	CALLS	#6, SMG\$\$CREATE_VIRTUAL_DISPLAY		
		58		50	D0	0005B	MOVL	R0, STATUS		
		5D		58	E9	0005E	BLBC	STATUS, 7\$	5742	
		50		08	BC	D0 00061	MOVL	@NEW_DISPLAY_ID, R0	5752	
	08	BC		38	A0	D1 00065	CMPL	56(R0), @NEW_DISPLAY_ID		
					06	12 0006A	BNEQ	3\$		
		11		44	A0	91 0006C	CMPB	68(R0), #17		
					04	13 00070	BEQL	4\$		
		50		59	D0	00072	MOVL	R9, R0		
					04	00075	RET			
		57		08	BC	D0 00076	MOVL	@NEW_DISPLAY_ID, DCB_NEW		
	38	A7		08	BC	D0 0007A	MOVL	@NEW_DISPLAY_ID, 56(DCB_NEW)	5757	
					07	11 0007F	BRB	6\$	5739	
	08	BC		50	D0	00081	MOVL	R0, @NEW_DISPLAY_ID	5763	
		57		50	D0	00085	MOVL	R0, DCB_NEW	5764	
10	B7	10		3C	A6	28 00088	MOVCL	60(DCB), @16(DCB), @16(DCB_NEW)	5774	
14	B7	14		3C	A6	28 0008F	MOVCL	60(DCB), @20(DCB), @20(DCB_NEW)	5778	
		50		02	A6	3C 00096	MOVZWL	2(DCB), R0	5783	
					50	D6 0009A	INCL	R0		
4C	B7	4C		50	28 0009C	MOVCL	R0, @76(DCB), @76(DCB_NEW)	5785		
		31		2F	A7	E9 000A2	BLBC	47(DCB_NEW), 8\$	5790	
		50		08	A6	9E 000A6	MOVAB	8(R6), DESC	5798	
				04	A0	D5 000AA	TSTL	4(DESC)	5803	
					28	13 000AD	BEQL	8\$		
				08	A7	9F 000AF	PUSHAB	8(DCB_NEW)	5807	
					50	DD 000B2	PUSHL	DESC		
		00000000G		02	FB	000B4	CALLS	#2, LIB\$SCOPY_DXDX		
				50	D0	000BB	MOVL	R0, STATUS		
		2C		58	E9	000BE	BLBC	STATUS, 9\$		
		31		2C	A6	B0 000C1	MOVW	44(DCB), 44(DCB_NEW)	5811	
				31	A6	B0 000C6	MOVW	49(DCB), 49(DCB_NEW)	5812	
					02	EF 000CB	EXTZV	#2, #1, 52(DCB), R0	5814	
34	50	34	A6	50	F0	000D1	INSV	R0, #2, #1, 52(DCB_NEW)		
	A7		01	18	A7	D5 000D7	TSTL	24(DCB_NEW)	5822	
			02	1E	13 000DA	BEQL	11\$			



				18	A7	9F	000DC		PUSHAB	24(DCB_NEW)	:	5826
				3C	A6	9F	000DF		PUSHAB	60(DCB)	:	5825
		00000000G	00		02	FB	000E2		CALLS	#2, LIB\$GET_VM	:	5826
			58		50	D0	000E9		MOVL	R0, STATUS	:	
			04		58	E8	000EC		BLBS	STATUS, 10\$	:	
			50		58	D0	000EF	9\$:	MOVL	STATUS, R0	:	5828
						04	000F2		RET		:	
18	B7				A6	28	000F3	10\$:	MOV C3	60(DCB), a24(DCB), a24(DCB_NEW)	:	5832
					A6	D0	000FA	11\$:	MOVL	40(DCB), 40(DCB_NEW)	:	5839
					01	D0	000FF		MOVL	#1, R0	:	5842
						04	00102		RET		:	5843

; Routine Size: 259 bytes,      Routine Base: \_SMG\$CODE + 1E6B

; 5615                    5844 1 !<BLF/PAGE>



```

: 5617 5845 1 %SBTTL 'SMG$$LOCATE_PP - Locate Pasting packet for given display and pasteboard'
: 5618 5846 1 GLOBAL ROUTINE SMG$$LOCATE_PP ( DCB : REF $DCB_DECL,
: 5619 5847 1 PBCB : REF $PBCB_DECL,
: 5620 5848 1 PP ) =
: 5621 5849 1 ++
: 5622 5850 1 FUNCTIONAL DESCRIPTION:
: 5623 5851 1
: 5624 5852 1     Locate the address of the pasting packet that joins this
: 5625 5853 1     virtual display to this pasteboard.
: 5626 5854 1
: 5627 5855 1 CALLING SEQUENCE:
: 5628 5856 1
: 5629 5857 1     ret_status.wlc.v = SMG$$LOCATE_PP (      DCB.rab.r,
: 5630 5858 1                                           PBCB.rab.r,
: 5631 5859 1                                           PP.wl.r)
: 5632 5860 1
: 5633 5861 1 FORMAL PARAMETERS:
: 5634 5862 1
: 5635 5863 1     DCB.rab.r      Address of a virtual display control block.
: 5636 5864 1
: 5637 5865 1     PBCB.rab.r     Address of a pasteboard control block.
: 5638 5866 1
: 5639 5867 1     PP.wl.r        Return address of the pasting packet that
: 5640 5868 1                     represents the pasting of the given virtual
: 5641 5869 1                     display to the given pasteboard control block.
: 5642 5870 1
: 5643 5871 1 IMPLICIT INPUTS:
: 5644 5872 1
: 5645 5873 1     None
: 5646 5874 1
: 5647 5875 1 IMPLICIT OUTPUTS:
: 5648 5876 1
: 5649 5877 1     None
: 5650 5878 1
: 5651 5879 1 COMPLETION STATUS:
: 5652 5880 1
: 5653 5881 1     SSS_NORMAL      Normal successful completion
: 5654 5882 1     SMG$_NOTPASTED  Given display is not pasted to given pasteboard
: 5655 5883 1
: 5656 5884 1 SIDE EFFECTS:
: 5657 5885 1
: 5658 5886 1     NONE
: 5659 5887 1 --
: 5660 5888 2 BEGIN
: 5661 5889 2 LOCAL
: 5662 5890 2     SEARCH_DCB : REF $DCB_DECL,      ! Addr of the DCB we'll actually
: 5663 5891 2                                     ! search for
: 5664 5892 2
: 5665 5893 2     CURR_PP : REF $PP_DECL;           ! Addr of pasting packet being
: 5666 5894 2                                     ! inspected.
: 5667 5895 2
: 5668 5896 2     CURR_PP = .DCB [DCB_A_PP_NEXT];   ! Start with 1st PP in chain
: 5669 5897 2
: 5670 5898 2 !+
: 5671 5899 2 ! If the virtual display is currently batched, the batch level will be non-zero.
: 5672 5900 2 ! This means a match needs to be found on the backup DCB address instead of the
: 5673 5901 2 ! DCB address.

```



```

5674 5902 2 !-
5675 5903 2 SEARCH_DCB = .DCB;
5676 5904 2
5677 5905 2 IF .DCB [DCB_L_BATCH_LEVEL] NEQ 0 ! Currently batched
5678 5906 2 THEN
5679 5907 2 SEARCH_DCB = .DCB [DCB_A_BACKUP_DCB];
5680 5908 2
5681 5909 2 WHILE .CURR_PP NEQ DCB [DCB_A_PP_NEXT] ! While any remain
5682 5910 2 DO
5683 5911 2 BEGIN ! Search for packet with matching PBCB addr
5684 5912 2 IF .CURR_PP [PP_A_DCB_ADDR] EQL .SEARCH_DCB AND
5685 5913 2 .CURR_PP [PP_A_PBCB_ADDR] EQL .PBCB
5686 5914 2 THEN
5687 5915 2 BEGIN ! Desired packet found
5688 5916 2 .PP = .CURR_PP;
5689 5917 2 RETURN (SS$_NORMAL); ! Return success
5690 5918 2 END; ! Desired packet found
5691 5919 2
5692 5920 2 CURR_PP = .CURR_PP [PP_A_NEXT_DCB]; ! Otherwise step along DCB
5693 5921 2 ! side of chain
5694 5922 2 END; ! Search for packet with matching PBCB addr
5695 5923 2
5696 5924 2
5697 5925 2
5698 5926 2 !+
5699 5927 2 If we fall out of the while loop, this virtual display is not pasted
5700 5928 2 to the specified pasteboard -- according to the pasting packets.
5701 5929 2
5702 5930 2 .PP = 0; ! To reduce liklihood someone will try to use it
5703 5931 2 ! and disregard status.
5704 5932 2 RETURN (SMG$_NOTPASTED); ! Return failure
5932 1 END; ! Routine SMG$$LOCATE_PP

```

			000C 00000	.ENTRY	SMG\$\$LOCATE_PP, Save R2,R3		5846
50	04	AC	D0 00002	MOVL	DCB, R0		5896
51	20	A0	D0 00006	MOVL	32(R0), CURR_PP		
53		50	D0 0000A	MOVL	R0, SEARCH_DCB		5903
	1C	A0	D5 0000D	TSTL	28(R0)		5905
		04	13 00010	BEQL	1\$		
53	40	A0	D0 00012	MOVL	64(R0), SEARCH_DCB		5907
52	20	A0	9E 00016 1\$:	MOVAB	32(R0), R2		5910
52		51	D1 0001A	CMPL	CURR_PP, R2		
		1A	13 0001D	BEQL	3\$		
53	10	A1	D1 0001F	CMPL	16(CURR_PP), SEARCH_DCB		5913
		0F	12 00023	BNEQ	2\$		
08	AC	A1	D1 00025	CMPL	20(CURR_PP), PBCB		5914
		08	12 0002A	BNEQ	2\$		
0C	BC	51	D0 0002C	MOVL	CURR_PP, @PP		5917
		01	D0 00030	MOVL	#1, R0		5918
		04	00033	RET			
51		61	D0 00034 2\$:	MOVL	(CURR_PP), CURR_PP		5921
		DD	11 00037	BRB	1\$		5910
	0C	BC	D4 00039 3\$:	CLRL	@PP		5929
50	00000000G	8F	D0 0003C	MOVL	#SMG\$_NOTPASTED, R0		5931



SMG\$DISPLAY\_LIN SMG\$DISPLAY\_LINKS - Virtual Display Linkages 16-Sep-1984 00:29:22  
1-096 SMG\$\$LOCATE\_PP - Locate Pasting packet for give 14-Sep-1984 13:09:43

04 00043

RET

VAX-11 Bliss-32 V4.0-742  
[SMGRTL.SRC]SMGDISLIN.B32;1

Page 165  
(34)

; 5932

; Routine Size: 68 bytes, Routine Base: \_SMG\$CODE + 1F6E

; 5705 5933 1 !<BLF/PAGE>



SMG\$DISPLAY_LIN	SMG\$DISPLAY_LINKS - Virtual Display Linkages	H 8	16-Sep-1984 00:29:22	VAX-11 Bliss-32 V4.0-742	Page 166
1-096	SMG\$\$PASTE_VIRTUAL_DISPLAY - Paste virtual disp		14-Sep-1984 13:09:43	[SMGRTL.SRC]SMGDISLIN.B32;1	(35)

```

5707 5934 1 %SBTTL 'SMG$$PASTE_VIRTUAL_DISPLAY - Paste virtual display to pasteboard'
5708 5935 1 GLOBAL ROUTINE SMG$$PASTE_VIRTUAL_DISPLAY (
5709 5936 1
5710 5937 1         DCB : REF $DCB DECL,
5711 5938 1         PBCB : REF $PBCB DECL,
5712 5939 1         PASTEBOARD_ROW,
5713 5940 1         PASTEBOARD_COL
5714 5941 1     ) =
5715 5942 1 ++
5716 5943 1     FUNCTIONAL DESCRIPTION:
5717 5944 1         The specified virtual display is "pasted" (oriented
5718 5945 1         with respect to) a pasteboard. This makes the display visible.
5719 5946 1         This is the inner paste routine. It assumes input parameters
5720 5947 1         are all present and valid. Further assumes that display
5721 5948 1         specified by DCB is not already pasted to pasteboard specified
5722 5949 1         by PBCB.
5723 5950 1
5724 5951 1     CALLING SEQUENCE:
5725 5952 1
5726 5953 1         ret_status.wlc.v = SMG$$PASTE_VIRTUAL_DISPLAY (
5727 5954 1             DCB.rab.r,
5728 5955 1             PBCB.rab.r,
5729 5956 1             PASTEBOARD_ROW.rl.r,
5730 5957 1             PASTEBOARD_COL.rl.r)
5731 5958 1
5732 5959 1     FORMAL PARAMETERS:
5733 5960 1
5734 5961 1         DCB.rab.r           Address of virtual display to be pasted.
5735 5962 1
5736 5963 1         PBCB.rab.r         Address of the pasteboard on
5737 5964 1                             which the pasting is to take place.
5738 5965 1
5739 5966 1         PASTEBOARD_ROW.rl.r Row on pasteboard which is to contain
5740 5967 1                             row 1 of the specified virtual display.
5741 5968 1
5742 5969 1         PASTEBOARD_COL.rl.r Column on pasteboard which is to contain
5743 5970 1                             column 1 of the specified virtual
5744 5971 1                             display.
5745 5972 1
5746 5973 1     IMPLICIT INPUTS:
5747 5974 1
5748 5975 1         None
5749 5976 1
5750 5977 1     IMPLICIT OUTPUTS:
5751 5978 1
5752 5979 1         None
5753 5980 1
5754 5981 1     COMPLETION STATUS:
5755 5982 1
5756 5983 1         $$$_NORMAL        Normal successful completion
5757 5984 1
5758 5985 1     SIDE EFFECTS:
5759 5986 1
5760 5987 1         NONE
5761 5988 1 --
5762 5989 2     BEGIN
5763 5990 2     LOCAL

```



```

5764 5991 2      STATUS,      ! Status of subroutine calls
5765 5992 2
5766 5993 2      PP      : REF $PP_DECL,      ! Addr of the pasting packet
5767 5994 2      ! being created.
5768 5995 2
5769 5996 2      WCB      : REF $WCB_DECL;      ! Addr. of window control block
5770 5997 2
5771 5998 2      !+
5772 5999 2      !- Get space for pasting packet.
5773 6000 2
5774 6001 2      IF NOT (STATUS = LIB$GET_VM ( %REF (PP_K_SIZE), PP))
5775 6002 2      THEN
5776 6003 2      RETURN (.STATUS);
5777 6004 2
5778 6005 2      CH$FILL (0, PP_K_SIZE, .PP);      ! Clear all fields to default 0
5779 6006 2
5780 6007 2      !+
5781 6008 2      !- Initialize pasting packet
5782 6009 2
5783 6010 2      PP [PP_A_DCB_ADDR] = .DCB;
5784 6011 2      PP [PP_A_PBCB_ADDR] = .PBCB;
5785 6012 2      PP [PP_W_ROW] = ..PASTEBOARD_ROW;
5786 6013 2      PP [PP_W_COL] = ..PASTEBOARD_COL;
5787 6014 2
5788 6015 2      !+
5789 6016 2      !- Plug this packet onto both queues.
5790 6017 2
5791 6018 2      $SMG$INSERT_AT_HEAD ( PP [PP_A_NEXT_DCB], DCB [DCB_A_PP_NEXT]);
5792 6019 2      $SMG$INSERT_AT_HEAD ( PP [PP_A_NEXT_PBCB], PBCB [PBCB_A_PP_NEXT]);
5793 6020 2
5794 6021 2      !+
5795 6022 2      !- If the display is batched, we want the backpointer in the PP to be
5796 6023 2      ! pointing to our backup DCB.
5797 6024 2
5798 6025 2      IF .DCB [DCB_L_BATCH_LEVEL] NEQ 0
5799 6026 2      THEN
5800 6027 2      PP [PP_A_DCB_ADDR] = .DCB [DCB_A_BACKUP_DCB];
5801 6028 2
5802 6029 2      !+
5803 6030 2      !- Recalc. occlusions introduced by this new pasting.
5804 6031 2
5805 6032 2      IF NOT ( STATUS = SMG$CHECK_OCCLUSION_FIRST ( .PBCB))
5806 6033 2      THEN
5807 6034 2      RETURN (.STATUS);
5808 6035 2
5809 6036 2      !+
5810 6037 2      !- Calculate the transformation constants needed to copy this display's
5811 6038 2      ! buffers into the associated window's buffers.
5812 6039 2
5813 6040 2      IF NOT ( STATUS = SMG$CALC_PASTE_TRANSF (.PP))
5814 6041 2      THEN
5815 6042 2      RETURN (.STATUS);
5816 6043 2
5817 6044 2      !+
5818 6045 2      !- If pasteboard batching enabled, quit here.
5819 6046 2
5820 6047 2      IF .PBCB [PBCB_L_BATCH_LEVEL] NEQ 0

```



00FC 00000	.ENTRY	SMG\$\$\$PASTE_VIRTUAL_DISPLAY, Save R2,R3,R4,-	5935
C2 00002	SUBL2	R5,R6,R7	:
9F 00005	PUSHAB	#8, SP	:
		PP	6001



37	00	04	AE	37	D0	00008	MOVL	#55, 4(SP)	6005
			00	AE	9F	0000C	PUSHAB	4(SP)	
			57	02	FB	0000F	CALLS	#2, LIB\$GET_VM	
			5B	50	D0	00016	MOVL	R0, STATUS	
			56	57	E9	00019	BLBC	STATUS, 2\$	
			6E	AE	D0	0001C	MOVL	PP, R6	
				00	2C	00020	MOVCS	#0, (SP), #0, #55, (R6)	
				66		00025			
			10	AC	7D	00026	MOVQ	DCB, 16(R6)	6010
			18	BC	B0	0002B	MOVW	@PASTEBOARD_ROW, 24(R6)	6012
			1A	BC	B0	00030	MOVW	@PASTEBOARD_COL, 26(R6)	6013
	50	04	AC	20	C1	00035	ADDL3	#32, DCB, R0	6018
			60	66	0E	0003A	INSQUE	(R6), (R0)	
			50	AE	D0	0003D	MOVL	PP, R0	6019
		08	BC	08	A0	0E0041	INSQUE	8(R0), @PBCB	
			52	AC	D0	00046	MOVL	DCB, R2	6025
				1C	A2	D50004A	TSTL	28(R2)	
				09	13	0004D	BEQL	1\$	
			50	AE	D0	0004F	MOVL	PP, R0	6027
	10		A0	40	A2	D00053	MOVL	64(R2), 16(R0)	
			53	08	AC	D00058	MOVL	PBCB, R3	6032
					53	DD0005C	PUSHL	R3	
	FA33		CF	01	FB	0005E	CALLS	#1, SMG\$CHECK_OCCLUSION_FIRST	
			57	50	D0	00063	MOVL	R0, STATUS	
			0E	57	E9	00066	BLBC	STATUS, 2\$	
				AE	DD	00069	PUSHL	PP	6040
	F649		CF	01	FB	0006C	CALLS	#1, SMG\$CALC_PASTE_TRANSF	
			57	50	D0	00071	MOVL	R0, STATUS	
			04	57	E8	00074	BLBS	STATUS, 3\$	
			50	57	D0	00077	MOVL	STATUS, R0	6042
					04	0007A	RET		
			56	00A4	C3	D0007B	MOVL	164(R3), R6	6047
					7C	1200080	BNEQ	6\$	
			51	08	A3	D00082	MOVL	8(R3), WCB	6056
			55	04	AE	D00086	MOVL	PP, R5	6061
			54	04	AE	D0008A	MOVL	PP, R4	6062
				1C	A2	D50008E	TSTL	28(R2)	6058
					22	1200091	BNEQ	4\$	
			50	28	A2	3C00093	MOVZWL	40(R2), R0	6061
			57	18	A5	3200097	CVTWL	24(R5), R7	
			50		57	C00009B	ADDL2	R7, R0	
20	A1		50		01	A30009E	SUBW3	#1, R0, 32(WCB)	
			52	2A	A2	3C000A3	MOVZWL	42(R2), R2	6062
			50	1A	A4	32000A7	CVTWL	26(R4), R0	
			52		50	C0000AB	ADDL2	R0, R2	
22	A1		52		01	A3000AE	SUBW3	#1, R2, 34(WCB)	
					24	11000B3	BRB	5\$	6058
			50	40	A2	D000B5	MOVL	64(R2), BACK_DCB	6071
			52	28	A0	3C000B9	MOVZWL	40(BACK_DCB), R2	6072
			57	18	A5	32000BD	CVTWL	24(R5), R7	
			52		57	C0000C1	ADDL2	R7, R2	
20	A1		52		01	A3000C4	SUBW3	#1, R2, 32(WCB)	
			50	2A	A0	3C000C9	MOVZWL	42(BACK_DCB), R0	6073
			52	1A	A4	32000CD	CVTWL	26(R4), R2	
			50		52	C0000D1	ADDL2	R2, R0	
22	A1		50		01	A3000D4	SUBW3	#1, R0, 34(WCB)	
					56	D5000D9	TSTL	R6	6080



00A8	C3		21	12	000DB	BNEQ	6\$		
00AA	C3	5F	01	B0	000DD	MOVW	#1, 168(R3)	:	6083
00AC	C3		A3	9B	000E2	MOVZBW	95(R3), 170(R3)	:	6084
00AE	C3	5A	01	B0	000E8	MOVW	#1, 172(R3)	:	6085
		04	A3	B0	000ED	MOVW	90(R3), 174(R3)	:	6086
00000000G	00		AE	DD	000F3	PUSHL	PP	:	6087
			01	FB	000F6	CALLS	#1, SMG\$\$FILL_WINDOW_BUFFER	:	
				04	000FD	RET		:	
	50	00000000G	8F	D0	000FE	MOVL	#SMG\$_BATWAS_ON, R0	:	6093
				04	00105	RET		:	6095

; Routine Size: 262 bytes,

Routine Base: \_SMG\$CODE + 1FB2

; 5869

6096 1 !<BLF/PAGE>



```

5871 6097 1 %SBTTL 'SMG$$RECALC_PP_FIELDS - Recalc. Pasting Packet fields'
5872 6098 1 GLOBAL ROUTINE SMG$$RECALC_PP_FIELDS (
5873 6099 1         DCB : REF $DCB_DECL
5874 6100 1     )=
5875 6101 1 ++
5876 6102 1 FUNCTIONAL DESCRIPTION:
5877 6103 1
5878 6104 1     This routine recalculates fields in the pasting packet that
5879 6105 1     need to change.
5880 6106 1     It walks the chain of pasting packets associated with the
5881 6107 1     given Display Control Block, updating each.
5882 6108 1
5883 6109 1 CALLING SEQUENCE:
5884 6110 1
5885 6111 1     ret_status.wlc.v = SMG$$RECALC_PP_FIELDS ( DCB.rab.r )
5886 6112 1
5887 6113 1 FORMAL PARAMETERS:
5888 6114 1
5889 6115 1     DCB.rab.r      Address of a virtual display control block.
5890 6116 1
5891 6117 1 IMPLICIT INPUTS:
5892 6118 1
5893 6119 1     None
5894 6120 1
5895 6121 1 IMPLICIT OUTPUTS:
5896 6122 1
5897 6123 1     None
5898 6124 1
5899 6125 1 COMPLETION STATUS:
5900 6126 1
5901 6127 1     SSS_NORMAL      Normal successful completion
5902 6128 1     Statuses returned by SMG$$CHECK_OCCLUSION and
5903 6129 1                        SMG$$CALC_PASTE_TRANF
5904 6130 1
5905 6131 1 SIDE EFFECTS:
5906 6132 1
5907 6133 1     NONE
5908 6134 1
5909 6135 2 --
5910 6136 2 BEGIN
5911 6137 2 LOCAL
5912 6138 2     PP : REF $PP_DECL;          ! Addr. of a pasting packet
5913 6139 2
5914 6140 2 ++ Step through all associated pasting packets, updating each.
5915 6141 2 --
5916 6142 2     PP = .DCB [DCB_A_PP_NEXT]; ! get 1st packet in DCB-oriented chain
5917 6143 2     WHILE .PP NEQ DCB [DCB_A_PP_NEXT] ! While any remain...
5918 6144 2     DO
5919 6145 2         BEGIN
5920 6146 2         LOCAL
5921 6147 2             STATUS,          ! Status of subroutine calls
5922 6148 2             PBCB : REF $PBCB_DECL; ! Addr of pasteboard control blk
5923 6149 2
5924 6150 2             PBCB = .PP [PP_A_PBCB_ADDR];
5925 6151 2             ++
5926 6152 2             ! Calculate who occludes who in current pasting chain.
5927 6153 2             --
  
```



```

: 5928      6154  4      IF NOT (STATUS =SMG$$CHECK_OCCLUSION ( .PBCB)) ! Recalc. occlusion
: 5929      6155  4      THEN
: 5930      6156  4      RETURN (.STATUS);
: 5931      6157  4
: 5932      6158  4      +
: 5933      6159  4      | Calculate critical constants used to map virtual displays
: 5934      6160  4      | to their correct position within the pasteboard buffer.
: 5935      6161  4      -
: 5936      6162  4      IF NOT ( STATUS = SMG$$CALC_PASTE_TRANSF ( .PP)) ! Clean up packet
: 5937      6163  4      THEN
: 5938      6164  4      RETURN (.STATUS);
: 5939      6165  4
: 5940      6166  4      PP = .PP [PP_A_NEXT_DCB];          ! Step to next packet
: 5941      6167  4      END;
: 5942      6168  4
: 5943      6169  4      RETURN ( SSS_NORMAL);
: 5944      6170  1      END;

```

! Routine SMG\$\$RECALC\_PP\_FIELDS

			000C 00000	.ENTRY	SMG\$\$RECALC_PP_FIELDS, Save R2,R3		6098
	52	04	AC D0 00002	MOVL	DCB, R2		6142
	53	20	A2 D0 00006	MOVL	32(R2), PP		
	50	20	A2 9E 0000A 1\$:	MOVAB	32(R2), R0		6143
	50		53 D1 0000E	CMPL	PP, R0		
			1D 13 00011	BEQL	2\$		
	50	14	A3 D0 00013	MOVL	20(PP), PBCB		6150
			50 DD 00017	PUSHL	PBCB		6154
F885	CF		01 FB 00019	CALLS	#1, SMG\$\$CHECK_OCCLUSION		
	12		50 E9 0001E	BLBC	STATUS, 3\$		
			53 DD 00021	PUSHL	PP		6162
F58C	CF		01 FB 00023	CALLS	#1, SMG\$\$CALC_PASTE_TRANSF		
	08		50 E9 00028	BLBC	STATUS, 3\$		
	53		63 D0 0002B	MOVL	(PP), PP		6166
			DA 11 0002E	BRB	1\$		6143
	50		01 D0 00030 2\$:	MOVL	#1, R0		6169
			04 00033 3\$:	RET			6170

; Routine Size: 52 bytes, Routine Base: \_SMG\$CODE + 20B8

; 5945 6171 1 !<BLF/PAGE>



```

5947 6172 1 %SBTTL 'SMG$$UNPASTE_VIRTUAL_DISPLAY - Unpaste virtual display from pasteboard'
5948 6173 1 GLOBAL ROUTINE SMG$$UNPASTE_VIRTUAL_DISPLAY (
5949 6174 1
5950 6175 1         DCB : REF $DCB_DECL,
5951 6176 1         PBCB : REF $PBCB_DECL
5952 6177 1     ) =
5953 6178 1
5954 6179 1 ++
5955 6180 1 FUNCTIONAL DESCRIPTION:
5956 6181 1
5957 6182 1     The specified virtual display is "unpasted" from a pasteboard
5958 6183 1     if a pasting packet can be found.
5959 6184 1     This is the inner-most unpasting routine. It assumes both
5960 6185 1     parameters are present and that they are valid (not necessarily
5961 6186 1     that the pasting packet which joins these two exists).
5962 6187 1
5963 6188 1 CALLING SEQUENCE:
5964 6189 1
5965 6190 1     ret_status.wlc.v = SMG$$UNPASTE_VIRTUAL_DISPLAY (
5966 6191 1         DCB.rab.r,
5967 6192 1         PBCB.rab.r)
5968 6193 1
5969 6194 1 FORMAL PARAMETERS:
5970 6195 1
5971 6196 1     DCB.rab.r           Address of DCB of virtual display to be
5972 6197 1                       unpasted.
5973 6198 1
5974 6199 1     PBCB.rab.r         Address of the pasteboard from
5975 6200 1                       which the unpasting is to take place.
5976 6201 1
5977 6202 1 IMPLICIT INPUTS:
5978 6203 1
5979 6204 1     None
5980 6205 1
5981 6206 1 IMPLICIT OUTPUTS:
5982 6207 1
5983 6208 1     None
5984 6209 1
5985 6210 1 COMPLETION STATUS:
5986 6211 1
5987 6212 1     SSS_NORMAL          Normal successful completion
5988 6213 1     SMG$_NOTPASTED      Specified virtual display is not currently
5989 6214 1                       pasted to the specified pasteboard.
5990 6215 1
5991 6216 1 SIDE EFFECTS:
5992 6217 1
5993 6218 1     NONE
5994 6219 1
5995 6220 1 --
5996 6221 1     BEGIN
5997 6222 1     LOCAL
5998 6223 1
5999 6224 1     STATUS,             ! Status of subroutine call
6000 6225 1     PP                  ! Addr of pasting packet being
6001 6226 1     : REF $PP_DECL;      ! inspected.
6002 6227 1
6003 6228 2
6004 6229 2
6005 6230 2
6006 6231 2
6007 6232 2
6008 6233 2
6009 6234 2
6010 6235 2
6011 6236 2
6012 6237 2
6013 6238 2
6014 6239 2
6015 6240 2
6016 6241 2
6017 6242 2
6018 6243 2
6019 6244 2
6020 6245 2
6021 6246 2
6022 6247 2
6023 6248 2
6024 6249 2
6025 6250 2
6026 6251 2
6027 6252 2
6028 6253 2
6029 6254 2
6030 6255 2
6031 6256 2
6032 6257 2
6033 6258 2
6034 6259 2
6035 6260 2
6036 6261 2
6037 6262 2
6038 6263 2
6039 6264 2
6040 6265 2
6041 6266 2
6042 6267 2
6043 6268 2
6044 6269 2
6045 6270 2
6046 6271 2
6047 6272 2
6048 6273 2
6049 6274 2
6050 6275 2
6051 6276 2
6052 6277 2
6053 6278 2
6054 6279 2
6055 6280 2
6056 6281 2
6057 6282 2
6058 6283 2
6059 6284 2
6060 6285 2
6061 6286 2
6062 6287 2
6063 6288 2
6064 6289 2
6065 6290 2
6066 6291 2
6067 6292 2
6068 6293 2
6069 6294 2
6070 6295 2
6071 6296 2
6072 6297 2
6073 6298 2
6074 6299 2
6075 6300 2
6076 6301 2
6077 6302 2
6078 6303 2
6079 6304 2
6080 6305 2
6081 6306 2
6082 6307 2
6083 6308 2
6084 6309 2
6085 6310 2
6086 6311 2
6087 6312 2
6088 6313 2
6089 6314 2
6090 6315 2
6091 6316 2
6092 6317 2
6093 6318 2
6094 6319 2
6095 6320 2
6096 6321 2
6097 6322 2
6098 6323 2
6099 6324 2
6100 6325 2
6101 6326 2
6102 6327 2
6103 6328 2
6104 6329 2
6105 6330 2
6106 6331 2
6107 6332 2
6108 6333 2
6109 6334 2
6110 6335 2
6111 6336 2
6112 6337 2
6113 6338 2
6114 6339 2
6115 6340 2
6116 6341 2
6117 6342 2
6118 6343 2
6119 6344 2
6120 6345 2
6121 6346 2
6122 6347 2
6123 6348 2
6124 6349 2
6125 6350 2
6126 6351 2
6127 6352 2
6128 6353 2
6129 6354 2
6130 6355 2
6131 6356 2
6132 6357 2
6133 6358 2
6134 6359 2
6135 6360 2
6136 6361 2
6137 6362 2
6138 6363 2
6139 6364 2
6140 6365 2
6141 6366 2
6142 6367 2
6143 6368 2
6144 6369 2
6145 6370 2
6146 6371 2
6147 6372 2
6148 6373 2
6149 6374 2
6150 6375 2
6151 6376 2
6152 6377 2
6153 6378 2
6154 6379 2
6155 6380 2
6156 6381 2
6157 6382 2
6158 6383 2
6159 6384 2
6160 6385 2
6161 6386 2
6162 6387 2
6163 6388 2
6164 6389 2
6165 6390 2
6166 6391 2
6167 6392 2
6168 6393 2
6169 6394 2
6170 6395 2
6171 6396 2
6172 6397 2
6173 6398 2
6174 6399 2
6175 6400 2
6176 6401 2
6177 6402 2
6178 6403 2
6179 6404 2
6180 6405 2
6181 6406 2
6182 6407 2
6183 6408 2
6184 6409 2
6185 6410 2
6186 6411 2
6187 6412 2
6188 6413 2
6189 6414 2
6190 6415 2
6191 6416 2
6192 6417 2
6193 6418 2
6194 6419 2
6195 6420 2
6196 6421 2
6197 6422 2
6198 6423 2
6199 6424 2
6200 6425 2
6201 6426 2
6202 6427 2
6203 6428 2
6204 6429 2
6205 6430 2
6206 6431 2
6207 6432 2
6208 6433 2
6209 6434 2
6210 6435 2
6211 6436 2
6212 6437 2
6213 6438 2
6214 6439 2
6215 6440 2
6216 6441 2
6217 6442 2
6218 6443 2
6219 6444 2
6220 6445 2
6221 6446 2
6222 6447 2
6223 6448 2
6224 6449 2
6225 6450 2
6226 6451 2
6227 6452 2
6228 6453 2
6229 6454 2
6230 6455 2
6231 6456 2
6232 6457 2
6233 6458 2
6234 6459 2
6235 6460 2
6236 6461 2
6237 6462 2
6238 6463 2
6239 6464 2
6240 6465 2
6241 6466 2
6242 6467 2
6243 6468 2
6244 6469 2
6245 6470 2
6246 6471 2
6247 6472 2
6248 6473 2
6249 6474 2
6250 6475 2
6251 6476 2
6252 6477 2
6253 6478 2
6254 6479 2
6255 6480 2
6256 6481 2
6257 6482 2
6258 6483 2
6259 6484 2
6260 6485 2
6261 6486 2
6262 6487 2
6263 6488 2
6264 6489 2
6265 6490 2
6266 6491 2
6267 6492 2
6268 6493 2
6269 6494 2
6270 6495 2
6271 6496 2
6272 6497 2
6273 6498 2
6274 6499 2
6275 6500 2
6276 6501 2
6277 6502 2
6278 6503 2
6279 6504 2
6280 6505 2
6281 6506 2
6282 6507 2
6283 6508 2
6284 6509 2
6285 6510 2
6286 6511 2
6287 6512 2
6288 6513 2
6289 6514 2
6290 6515 2
6291 6516 2
6292 6517 2
6293 6518 2
6294 6519 2
6295 6520 2
6296 6521 2
6297 6522 2
6298 6523 2
6299 6524 2
6300 6525 2
6301 6526 2
6302 6527 2
6303 6528 2
6304 6529 2
6305 6530 2
6306 6531 2
6307 6532 2
6308 6533 2
6309 6534 2
6310 6535 2
6311 6536 2
6312 6537 2
6313 6538 2
6314 6539 2
6315 6540 2
6316 6541 2
6317 6542 2
6318 6543 2
6319 6544 2
6320 6545 2
6321 6546 2
6322 6547 2
6323 6548 2
6324 6549 2
6325 6550 2
6326 6551 2
6327 6552 2
6328 6553 2
6329 6554 2
6330 6555 2
6331 6556 2
6332 6557 2
6333 6558 2
6334 6559 2
6335 6560 2
6336 6561 2
6337 6562 2
6338 6563 2
6339 6564 2
6340 6565 2
6341 6566 2
6342 6567 2
6343 6568 2
6344 6569 2
6345 6570 2
6346 6571 2
6347 6572 2
6348 6573 2
6349 6574 2
6350 6575 2
6351 6576 2
6352 6577 2
6353 6578 2
6354 6579 2
6355 6580 2
6356 6581 2
6357 6582 2
6358 6583 2
6359 6584 2
6360 6585 2
6361 6586 2
6362 6587 2
6363 6588 2
6364 6589 2
6365 6590 2
6366 6591 2
6367 6592 2
6368 6593 2
6369 6594 2
6370 6595 2
6371 6596 2
6372 6597 2
6373 6598 2
6374 6599 2
6375 6600 2
6376 6601 2
6377 6602 2
6378 6603 2
6379 6604 2
6380 6605 2
6381 6606 2
6382 6607 2
6383 6608 2
6384 6609 2
6385 6610 2
6386 6611 2
6387 6612 2
6388 6613 2
6389 6614 2
6390 6615 2
6391 6616 2
6392 6617 2
6393 6618 2
6394 6619 2
6395 6620 2
6396 6621 2
6397 6622 2
6398 6623 2
6399 6624 2
6400 6625 2
6401 6626 2
6402 6627 2
6403 6628 2
6404 6629 2
6405 6630 2
6406 6631 2
6407 6632 2
6408 6633 2
6409 6634 2
6410 6635 2
6411 6636 2
6412 6637 2
6413 6638 2
6414 6639 2
6415 6640 2
6416 6641 2
6417 6642 2
6418 6643 2
6419 6644 2
6420 6645 2
6421 6646 2
6422 6647 2
6423 6648 2
6424 6649 2
6425 6650 2
6426 6651 2
6427 6652 2
6428 6653 2
6429 6654 2
6430 6655 2
6431 6656 2
6432 6657 2
6433 6658 2
6434 6659 2
6435 6660 2
6436 6661 2
6437 6662 2
6438 6663 2
6439 6664 2
6440 6665 2
6441 6666 2
6442 6667 2
6443 6668 2
6444 6669 2
6445 6670 2
6446 6671 2
6447 6672 2
6448 6673 2
6449 6674 2
6450 6675 2
6451 6676 2
6452 6677 2
6453 6678 2
6454 6679 2
6455 6680 2
6456 6681 2
6457 6682 2
6458 6683 2
6459 6684 2
6460 6685 2
6461 6686 2
6462 6687 2
6463 6688 2
6464 6689 2
6465 6690 2
6466 6691 2
6467 6692 2
6468 6693 2
6469 6694 2
6470 6695 2
6471 6696 2
6472 6697 2
6473 6698 2
6474 6699 2
6475 6700 2
6476 6701 2
6477 6702 2
6478 6703 2
6479 6704 2
6480 6705 2
6481 6706 2
6482 6707 2
6483 6708 2
6484 6709 2
6485 6710 2
6486 6711 2
6487 6712 2
6488 6713 2
6489 6714 2
6490 6715 2
6491 6716 2
6492 6717 2
6493 6718 2
6494 6719 2
6495 6720 2
6496 6721 2
6497 6722 2
6498 6723 2
6499 6724 2
6500 6725 2
6501 6726 2
6502 6727 2
6503 6728 2
6504 6729 2
6505 6730 2
6506 6731 2
6507 6732 2
6508 6733 2
6509 6734 2
6510 6735 2
6511 6736 2
6512 6737 2
6513 6738 2
6514 6739 2
6515 6740 2
6516 6741 2
6517 6742 2
6518 6743 2
6519 6744 2
6520 6745 2
6521 6746 2
6522 6747 2
6523 6748 2
6524 6749 2
6525 6750 2
6526 6751 2
6527 6752 2
6528 6753 2
6529 6754 2
6530 6755 2
6531 6756 2
6532 6757 2
6533 6758 2
6534 6759 2
6535 6760 2
6536 6761 2
6537 6762 2
6538 6763 2
6539 6764 2
6540 6765 2
6541 6766 2
6542 6767 2
6543 6768 2
6544 6769 2
6545 6770 2
6546 6771 2
6547 6772 2
6548 6773 2
6549 6774 2
6550 6775 2
6551 6776 2
6552 6777 2
6553 6778 2
6554 6779 2
6555 6780 2
6556 6781 2
6557 6782 2
6558 6783 2
6559 6784 2
6560 6785 2
6561 6786 2
6562 6787 2
6563 6788 2
6564 6789 2
6565 6790 2
6566 6791 2
6567 6792 2
6568 6793 2
6569 6794 2
6570 6795 2
6571 6796 2
6572 6797 2
6573 6798 2
6574 6799 2
6575 6800 2
6576 6801 2
6577 6802 2
6578 6803 2
6579 6804 2
6580 6805 2
6581 6806 2
6582 6807 2
6583 6808 2
6584 6809 2
6585 6810 2
6586 6811 2
6587 6812 2
6588 6813 2
6589 6814 2
6590 6815 2
6591 6816 2
6592 6817 2
6593 6818 2
6594 6819 2
6595 6820 2
6596 6821 2
6597 6822 2
6598 6823 2
6599 6824 2
6600 6825 2
6601 6826 2
6602 6827 2
6603 6828 2
6604 6829 2
6605 6830 2
6606 6831 2
6607 6832 2
6608 6833 2
6609 6834 2
6610 6835 2
6611 6836 2
6612 6837 2
6613 6838 2
6614 6839 2
6615 6840 2
6616 6841 2
6617 6842 2
6618 6843 2
6619 6844 2
6620 6845 2
6621 6846 2
6622 6847 2
6623 6848 2
6624 6849 2
6625 6850 2
6626 6851 2
6627 6852 2
6628 6853 2
6629 6854 2
6630 6855 2
6631 6856 2
6632 6857 2
6633 6858 2
6634 6859 2
6635 6860 2
6636 6861 2
6637 6862 2
6638 6863 2
6639 6864 2
6640 6865 2
6641 6866 2
6642 6867 2
6643 6868 2
6644 6869 2
6645 6870 2
6646 6871 2
6647 6872 2
6648 6873 2
6649 6874 2
6650 6875 2
6651 6876 2
6652 6877 2
6653 6878 2
6654 6879 2
6655 6880 2
6656 6881 2
6657 6882 2
6658 6883 2
6659 6884 2
6660 6885 2
6661 6886 2
6662 6887 2
6663 6888 2
6664 6889 2
6665 6890 2
6666 6891 2
6667 6892 2
6668 6893 2
6669 6894 2
6670 6895 2
6671 6896 2
6672 6897 2
6673 6898 2
6674 6899 2
6675 6900 2
6676 6901 2
6677 6902 2
6678 6903 2
6679 6904 2
6680 6905 2
6681 6906 2
6682 6907 2
6683 6908 2
6684 6909 2
6685 6910 2
6686 6911 2
6687 6912 2
6688 6913 2
6689 6914 2
6690 6915 2
6691 6916 2
6692 6917 2
6693 6918 2
6694 6919 2
6695 6920 2
6696 6921 2
6697 6922 2
6698 6923 2
6699 6924 2
6700 6925 2
6701 6926 2
6702 6927 2
6703 6928 2
6704 6929 2
6705 6930 2
6706 6931 2
6707 6932 2
6708 6933 2
6709 6934 2
6710 6935 2
6711 6936 2
6712 6937 2
6713 6938 2
6714 6939 2
6715 6940 2
6716 6941 2
6717 6942 2
6718 6943 2
6719 6944 2
6720 6945 2
6721 6946 2
6722 6947 2
6723 6948 2
6724 6949 2
6725 6950 2
6726 6951 2
6727 6952 2
6728 6953 2
6729 6954 2
6730 6955 2
6731 6956 2
6732 6957 2
6733 6958 2
6734 6959 2
6735 6960 2
6736 6961 2
6737 6962 2
6738 6963 2
6739 6964 2
6740 6965 2
6741 6966 2
6742 6967 2
6743 6968 2
6744 6969 2
6745 6970 2
6746 6971 2
6747 6972 2
6748 6973 2
6749 6974 2
6750 6975 2
6751 6976 2
6752 6977 2
6753 6978 2
6754 6979 2
6755 6980 2
6756 6981 2
6757 6982 2
6758 6983 2
6759 6984 2
6760 6985 2
6761 6986 2
6762 6987 2
6763 6988 2
6764 6989 2
6765 6990 2
6766 6991 2
6767 6992 2
6768 6993 2
6769 6994 2
6770 6995 2
6771 6996 2
6772 6997 2
6773 6998 2
6774 6999 2
6775 7000 2
6776 7001 2
6777 7002 2
6778 7003 2
6779 7004 2
6780 7005 2
6781 7006 2
6782 7007 2
6783 7008 2
6784 7009 2
6785 7010 2
6786 7011 2
6787 7012 2
6788 7013 2
6789 7014 2
6790 7015 2
6791 7016 2
6792 7017 2
6793 7018 2
6794 7019 2
6795 7020 2
6796 7021 2
6797 7022 2
6798 7023 2
6799 7024 2
6800 7025 2
6801 7026 2
6802 7027 2
6803 7028 2
6804 7029 2
6805 7030 2
6806 7031 2
6807 7032 2
6808 7033 2
6809 7034 2
6810 7035 2
6811 7036 2
6812 7037 2
6813 7038 2
6814 7039 2
6815 7040 2
6816 7041 2
6817 7042 2
6818 7043 2
6819 7044 2
6820 7045 2
6821 7046 2
6822 7047 2
6823 7048 2
6824 7049 2
6825 7050 2
6826 7051 2
6827 7052 2
6828 7053 2
6829 7054 2
6830 7055 2
6831 7056 2
6832 7057 2
6833 7058 2
6834 7059 2
6835 7060 2
6836 7061 2
6837 7062 2
6838 7063 2
6839 7064 2
6840 7065 2
6841 7066 2
6842 7067 2
6843 7068 2
6844 7069 2
6845 7070 2
6846 7071 2
6847 7072 2
6848 7073 2
6849 7074 2
6850 7075 2
6851 7076 2
6852 7077 2
6853 7078 2
6854 7079 2
6855 7080 2
6856 7081 2
6857 7082 2
6858 7083 2
6859 7084 2
6860 7085 2
6861 7086 2
6862 7087 2
6863 7088 2
6864 7089 2
6865 7090 2
6866 7091 2
6867 7092 2
6868 7093 2
6869 7094 2
6870 7095 2
6871 7096 2
6872 7097 2
6873 7098 2
6874 7099 2
6875 7100 2
6876 7101 2
6877 7102 2
6878 7103 2
6879 7104 2
6880 7105 2
6881 7106 2
6882 7107 2
6883 7108 2
6884 7109 2
6885 7110 2
6886 7111 2
6887 7112 2
6888 7113 2
6889 7114 2
6890 7115 2
6891 7116 2
6892 7117 2
6893 7118 2
6894 7119 2
6895 7120 2
6896 7121 2
6897 7122 2
6898 7123 2
6899 7124 2
6900 7125 2
6901 7126 2
6902 7127 2
6903 7128 2
6904 7129 2
6905 7130 2
6906 7131 2
6907 7132 2
6908 7133 2
6909 7134 2
6910 7135 2
6911 7136 2
6912 7137 2
6913 7138 2
6914 7139 2
6915 7140 2
6916 7141 2
6917 7142 2
6918 7143 2
6919 7144 2
6920 7145 2
6921 7146 2
6922 7147 2
6923 7148 2
6924 7149 2
6925 7150 2
6926 7151 2
6927 7152 2
6928 7153 2
6929 7154 2
6930 7155 2
6931 7156 2
6932 7157 2
6933 7158 2
6934 7159 2
6935 71
```



```

: 6004      6229      3      IF NOT (STATUS = SMG$$LOCATE_PP ( .DCB, .PBCB, PP))
: 6005      6230      3      THEN
: 6006      6231      3      RETURN (.STATUS);      ! No common pasting packet exists
: 6007      6232      3
: 6008      6233      3      !+
: 6009      6234      3      Located desired packet. Remove it from both queues.
: 6010      6235      3      !-
: 6011      6236      3      SSMG$REMOVE_FROM_QUEUE ( PP [PP_A_NEXT_DCB] );
: 6012      6237      3      SSMG$REMOVE_FROM_QUEUE ( PP [PP_A_NEXT_PBCB] );
: 6013      6238      3
: 6014      6239      3      !+
: 6015      6240      3      Give back the pasting packet space
: 6016      6241      3      !-
: 6017      6242      3      IF NOT (STATUS = LIB$FREE_VM ( %REF(PP_K_SIZE), PP))
: 6018      6243      3      THEN
: 6019      6244      3      RETURN (.STATUS);
: 6020      6245      3
: 6021      6246      3      !+
: 6022      6247      3      If other virtual displays are still pasted to this pasteboard, we need
: 6023      6248      3      to recalculate their occlusion bits since they may have changed by
: 6024      6249      3      removing this virtual display.
: 6025      6250      3      !-
: 6026      6251      3      IF .PBCB [ PBCB_A_PP_NEXT] NEQ PBCB [ PBCB_A_PP_NEXT]
: 6027      6252      3      THEN
: 6028      6253      3      IF NOT ( STATUS = SMG$$CHECK_OCCLUSION ( .PBCB ))
: 6029      6254      3      THEN
: 6030      6255      3      RETURN (.STATUS);
: 6031      6256      3
: 6032      6257      3      !+
: 6033      6258      3      Cause pasteboard to reflect this change.
: 6034      6259      3      !-
: 6035      6260      3
: 6036      6261      3      RETURN ( SMG$$CHECK_FOR_OUTPUT_PBCB ( .PBCB ));
: 6037      6262      3
: 6038      6263      3      END;

```

```

! Routine SMG$UNPASTE_VIRTUAL_DISPLAY

```

				0004 00000	.ENTRY	SMG\$UNPASTE_VIRTUAL_DISPLAY, Save R2	: 6173
	5E		04	08 C2 00002	SUBL2	#8, SP	
			04	AE 9F 00005	PUSHAB	PP	: 6229
	7E			AC 7D 00008	MOVQ	DCB, -(SP)	
	FE71	CF	04	03 FB 0000C	CALLS	#3, SMG\$\$LOCATE_PP	
		3C		50 E9 00011	BLBC	STATUS, 2\$	
		51	04	BE 0F 00014	REMQUE	@PP, F00	: 6236
51	04	AE		08 C1 00018	ADDL3	#8, PP, R1	: 6237
		52		61 0F 0001D	REMQUE	(R1), F00	
			04	AE 9F 00020	PUSHAB	PP	: 6242
	04	AE		37 D0 00023	MOVL	#55, 4(SP)	
			04	AE 9F 00027	PUSHAB	4(SP)	
00000000G	00			02 FB 0002A	CALLS	#2, LIB\$FREE_VM	
	1C			50 E9 00031	BLBC	STATUS, 2\$	
08	AC		08	BC D1 00034	CMPL	@PBCB, PBCB	: 6251
			08	0B 13 00039	BEQL	1\$	
				AC DD 0003B	PUSHL	PBCB	: 6253



SMG\$DISPLAY_LIN	SMG\$DISPLAY LINKS - Virtual Display Linkages	D 9	16-Sep-1984 00:29:22	VAX-11 Bliss-32 V4.0-742	Page 175
1-096	SMG\$\$UNPASTE_VIRTUAL_DISPLAY - Unpaste virtual		14-Sep-1984 13:09:43	[SMGRTL.SRC]SMGDISLIN.B32;1	(37)

F82C	CF	01	FB 0003E	CALLS	#1, SMG\$\$CHECK_OCCLUSION	:
	0A	50	E9 00043	BLBC	STATUS, 2\$	:
		08	AC DD 00046	PUSHL	PBCB	: 6261
00000000G	00	01	FB 00049	CALLS	#1, SMG\$\$CHECK_FOR_OUTPUT_PBCB	:
			04 00050	RET		: 6263

; Routine Size: 81 bytes,      Routine Base: \_SMG\$CODE + 20EC

; 6039                      6264 1 !<BLF/PAGE>



SMG\$DISPLAY\_LIN

1-096

SMG\$DISPLAY LINKS - Virtual Display Linkages

SMG\$\$UNPASTE\_VIRTUAL\_DISPLAY - Unpaste virtual

E 9

16-Sep-1984 00:29:22

14-Sep-1984 13:09:43

VAX-11 Bliss-32 V4.0-742

[SMGRTL.SRC]SMGDISLIN.B32;1

Page 176

(38)

: 6041

: 6042

: 6043

6265 1 END

6266 1

6267 0 ELUDOM

! End of module SMG\$DISPLAY\_LINKS

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
SMG\$DATA	80	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2)
SMG\$CODE	8509	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	101	1	581	00:00.9
\$255\$DUA28:[SMGRTL.OBJ]RTLLIB.L32;1	36	0	0	8	00:00.1
\$255\$DUA28:[SMGRTL.OBJ]SMGLIB.L32;1	469	152	32	38	00:00.4

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:SMGDISLIN/OBJ=OBJ\$:SMGDISLIN MSRC\$:SMGDISLIN/UPDATE=(ENH\$:SMGDISLIN

)

Size: 8471 code + 118 data bytes

Run Time: 02:45.3

Elapsed Time: 08:00.6

Lines/CPU Min: 2275

Lexemes/CPU-Min: 20356

Memory Used: 435 pages

Compilation Complete



0356 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY



0357 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

